This Page Is Inserted by IFW Operations
and is not a part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

As rescanning documents *will not* correct images, please do not report the images to the Image Problem Mailbox.

Refine Search

Search Results -

Terms	Documents
L5 and JIT	14

Database:
US Pre-Grant Publication Full-Text Database
US Patents Full-Text Database
US OCR Full-Text Database
EPO Abstracts Database
JPO Abstracts Database
Derwent World Patents Index
IBM Technical Disclosure Bulletins

Search: Search and PLUS RESULTS

Refine Search

Recall Text | Clear | Interrupt

Search History

DATE: Saturday, May 29, 2004 Printable Copy Create Case

Set Name side by side	Query	Hit Count	Set Name result set
	DB=USPT; PLUR=NO; OP=OR		
L8	L5 and JIT	14	L8
L7	L5 NOT l4	49	L7
L6	L5 NOT l1	34	L6
L5	(6253215 6175956 6382846 5598560 5247679 5301325 5307492 5339238 6075942 5953741 5857104 6243857 5966537 5797014 5845118 5752014 5590331 6131187 5999732 5761513 5784553 5923892 5937193 6085307 6233733 6298477 6381737 6412109 5185868 5251306 5903758 5339420 5465258 6047125 6192517 6112025 6338160 5937411 5526499 5452457 5381547 6118940 6093216 5396631 6385764 5974529 6182107 6282702 6412021 6557023).pn.	50	L5
L4	L3 and (dependency OR dependencies)	58	L4
L3	L1 and JAVA	337	L3
L2	L1 and compiler	353	L2
L1	JIT OR (just ADJ in adj time) OR (just-in-time) AND (compiler OR compiling)	672	L1

END OF SEARCH HISTORY

Hit List

Clear | Generate Collection | Print | Fwd Refs | Bkwd Refs | Generate OACS

Search Results - Record(s) 1 through 50 of 58 returned.

☐ 1. Document ID: US 6738967 B1

L4: Entry 1 of 58 File: USPT May 18, 2004

US-PAT-NO: 6738967
DOCUMENT-IDENTIFIER: US 6738967 B1

TITLE: Compiling for multiple virtual machines targeting different processor architectures

DATE-ISSUED: May 18, 2004

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Radigan; James J.	Redmond	WA		

US-CL-CURRENT: 717/146

ABSTRACT:

A developer's system compiles a source-language program by parsing it into an intermediate language (IL) program that is independent of the architecture or resources of any particular processor. This system generates a set of machine independent annotations that record attributes and structure of the IL program such as variable definitions and uses. The annotations are in the form of a unique graph structure. Further annotations are used to associate the machine specific optimization information with each of many different microprocessors. The IL program and all the annotations are distributed to multiple user systems having mutually different virtual machines that translate the IL program into object language for their respective different processors. Optimizers in the virtual machines select graph edges based on the machine specific annotations to generate an optimized object program. The translators are table-driven. Entities in the IL program are keyed to entries in a replaceable table in each virtual machine.

74 Claims, 12 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 10

Full | Title | Citation | Front | Review | Classification | Date | Reference | Claims | KWIC | Draw De

☐ 2. Document ID: US 6732363 B1

L4: Entry 2 of 58 File: USPT May 4, 2004

US-PAT-NO: 6732363
DOCUMENT-IDENTIFIER: US 6732363 B1

TITLE: Supporting inter-process communication through a conditional trap instruction

DATE-ISSUED: May 4, 2004

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Chaudhry; Shailender	San Francisco	CA		
Tremblay; Marc	Menlo Park	CA		

US-CL-CURRENT: 719/318; 719/310

ABSTRACT:

One embodiment of the present invention provides a system that supports inter-process communication through use of a conditional trap instruction. The system operates by allowing a first process to send a communication to a second process by writing to a register that is visible to the second process. The second process then examines a value in the register by executing the conditional trap instruction. If the value in the register satisfies a condition specified by the conditional trap instruction, the system executes a trap handling routine that takes an action in response to the communication from the first process. If the value in the register does not satisfy the condition, the system takes no action and proceeds with execution of the code. In one embodiment of the present invention, the first process writes to the register by causing an interrupt that writes to the register.

27 Claims, 17 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 7



☐ 3. Document ID: US 6721944 B2

L4: Entry 3 of 58 File: USPT Apr 13, 2004

US-PAT-NO: 6721944
DOCUMENT-IDENTIFIER: US 6721944 B2

TITLE: Marking memory elements based upon usage of accessed information during speculative execution

DATE-ISSUED: April 13, 2004

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Chaudhry; Shailender	San Francisco	CA		
Tremblay; Marc	Menlo Park	CA		

US-CL-CURRENT: 717/154; 712/228, 717/140, 718/102

ABSTRACT:

One embodiment of the present invention provides a system that marks memory elements based upon how information retrieved from the memory elements affects speculative program execution. This system operates by allowing a programmer to examine source code that is to be compiled into executable code for a head thread that executes program instructions, and for a speculative thread that executes program instructions in advance of the head thread. During read operations to memory elements by the speculative thread, this executable code generally causes the speculative thread to update status information associated with the memory elements to indicate that the memory elements have been read by the speculative thread. Next, the system allows the programmer to identify a given read operation directed to a given memory element, wherein a given value retrieved from the given memory element during the given read operation does not affect subsequent execution of the speculative thread. The programmer is then allowed to insert a hint into the source code specifying that the speculative thread is not to update status information during the given read operation directed to the given memory element. Next, the system compiles the source code, including the hint, into the executable code, so that during the given read operation, the executable code does not cause the speculative thread to update status information associated with the given memory element to indicate that the given memory element has been read by the speculative thread.

24 Claims, 14 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 6



☐ 4. Document ID: US 6721713 B1

L4: Entry 4 of 58 File: USPT Apr 13, 2004

US-PAT-NO: 6721713
DOCUMENT-IDENTIFIER: US 6721713 B1

TITLE: Business alliance identification in a web architecture framework

DATE-ISSUED: April 13, 2004

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Guheen; Michael F.	Tiburon	CA		
Mitchell; James D.	Manhattan Beach	CA		
Barrese; James J.	San Jose	CA		

US-CL-CURRENT: 705/1; 709/223, 715/503

ABSTRACT:

A system, method and article of manufacture are provided for identifying alliances among a plurality of business entities in components of a network framework. First,

alliances are identified among a plurality of business entities in terms of components of a current network framework. Next, a pictorial representation is displayed of the current network framework and the components. The alliances are then conveyed by indicia coding the components of the current network framework in which the alliances exist.

10 Claims, 177 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 177

☐ 5. Document ID: US 6718535 B1

L4: Entry 5 of 58 File: USPT Apr 6, 2004

US-PAT-NO: 6718535
DOCUMENT-IDENTIFIER: US 6718535 B1

TITLE: System, method and article of manufacture for an activity framework design in an e-commerce based environment

DATE-ISSUED: April 6, 2004

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Underwood; Roy Aaron	Long Grove	IL		

US-CL-CURRENT: 717/101; 717/120

ABSTRACT:

A system and method are provided for providing an activity framework. First, a plurality of sub-activities are created which each include sub-activity logic adapted to generate an output based on an input received from a user upon execution. Second, a plurality of activities are defined which each execute the sub-activities in a unique manner upon being selected for accomplishing a goal associated with the activity. Selection of one of the activities is allowed by receiving user indicia. An interface is depicted for allowing receipt of the input and display of the output during execution of the sub-activities associated with the selected activity.

24 Claims, 179 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 111

☐ 6. Document ID: US 6718457 B2

L4: Entry 6 of 58 File: USPT Apr 6, 2004

US-PAT-NO: 6718457
DOCUMENT-IDENTIFIER: US 6718457 B2

TITLE: Multiple-thread processor for threaded software applications

DATE-ISSUED: April 6, 2004

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Tremblay; Marc	Menlo Park	CA		
Joy; William	Aspen	CO		

US-CL-CURRENT: 712/212

ABSTRACT:

A processor has an improved architecture for multiple-thread operation on the basis of a highly parallel structure including multiple independent parallel execution paths for executing in parallel across threads and a multiple-instruction parallel pathway within a thread. The multiple independent parallel execution paths include functional units that execute an instruction set including special data-handling instructions that are advantageous in a multiple-thread environment.

35 Claims, 10 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 7

☐ 7. Document ID: US 6704927 B1

L4: Entry 7 of 58 File: USPT Mar 9, 2004

US-PAT-NO: 6704927
DOCUMENT-IDENTIFIER: US 6704927 B1

TITLE: Static binding of dynamically-dispatched calls in the presence of dynamic linking and loading

DATE-ISSUED: March 9, 2004

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Bak; Lars	Palo Alto	CA		
Mitrovic; Srdjan	Redwood Shores	CA		
Holzle; Urs	Goleta	CA		

US-CL-CURRENT: 717/151; 717/139, 717/140, 717/148, 717/152, 717/153, 717/165

ABSTRACT:

Techniques for performing static binding of dispatched-calls in the presence of

dynamic linking and loading are provided. A method for increasing the execution performance of a function at run-time includes compiling the function, which may either be interpreted or previously compiled, and identifying a call within the function to a process. The method also includes adding dependency information to the function. The dependency information is arranged to indicate a status of the function, and contains information pertaining to the class, the name, and the signature associated with the process. In one embodiment, the process is a virtual process, and the method includes analyzing a class structure associated with the function in order to determine when the virtual process is a substantially unique target of the call. In such an embodiment, the virtual process may be inlined into the function when it is determined that the virtual process is the substantially unique target of the call.

26 Claims, 8 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 7

☐ 8. Document ID: US 6704925 B1

L4: Entry 8 of 58 File: USPT Mar 9, 2004

US-PAT-NO: 6704925
DOCUMENT-IDENTIFIER: US 6704925 B1

TITLE: Dynamic binary translator with a system and method for updating and maintaining coherency of a translation cache

DATE-ISSUED: March 9, 2004

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Bugnion; Edouard	Menlo Park	CA		

US-CL-CURRENT: 717/138; 703/26, 718/1

ABSTRACT:

A dynamic binary translator converts input instruction sequences into output instruction sequences that are stored in a translation cache. In order to maintain coherence of the translation cache with the run-time version of the input instructions, translated code is checked by either a conflict detection mechanism or a code-invariance mechanism. For conflict detection, the system preferably uses memory traces generated by the memory management unit of the underlying hardware processor. In order to check for code-invariance, preludes for comparing cached, output instruction sequences with their supposed run-time input instruction equivalents are appended to the cached instructions themselves. Changes in the input sequences then result only in retranslation of instruction sequences in which at least one instruction has changed; this avoids costly total flushes of the translation cache. An additional prelude is appended to any cached output sequences displaying characteristics of potentially self-constant-modifying code. If the input instructions have since translation changed with respect only to constants, then the constants are updated before execution; this completely eliminates the

need for flushing the translation cache or retranslating the instruction if only constants have changed since the original translation. The invention is preferably incorporated in a virtual machine monitor, on which a virtual machine is running.

11 Claims, 3 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 3

Full | Title | Citation | Front | Review | Classification | Date | Reference | Claims | KWIC | Draw D

☐ 9. Document ID: US 6704873 B1

L4: Entry 9 of 58 File: USPT Mar 9, 2004

US-PAT-NO: 6704873
DOCUMENT-IDENTIFIER: US 6704873 B1

TITLE: Secure gateway interconnection in an e-commerce based environment

DATE-ISSUED: March 9, 2004

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Underwood; Roy Aaron	Long Grove	IL		

US-CL-CURRENT: 713/201; 709/223, 709/249

ABSTRACT:

A system and method of providing a global internetworking gateway architecture in an e-commerce environment are provided. A plurality of gateways each situated in a distinct geographic location are coupled to an internet. A wide area network, separate from the internet, is coupled to each of the gateways for providing communication between the wide area network and the internet. Coupled to the wide area network is a central database for providing a central storage for data used in e-commerce carried out over the internet. In one embodiment, at least one of the gateways includes at least one screening router coupled to the internet service provider, at least one firewall connected to the screening router, and a choker router coupled between the wide area network and the firewall.

16 Claims, 179 Drawing figures
Exemplary Claim Number: 5
Number of Drawing Sheets: 111

Full | Title | Citation | Front | Review | Classification | Date | Reference | Claims | KWIC | Draw D

☐ 10. Document ID: US 6704862 B1

L4: Entry 10 of 58 File: USPT Mar 9, 2004

US-PAT-NO: 6704862

DOCUMENT-IDENTIFIER: US 6704862 B1

TITLE: Method and apparatus for facilitating exception handling using a conditional trap instruction

DATE-ISSUED: March 9, 2004

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Chaudhry; Shailender	San Francisco	CA		
Tremblay; Marc	Menlo Park	CA		

US-CL-CURRENT: 712/244; 712/24

ABSTRACT:

One embodiment of the present invention provides a system that supports exception handling through use of a conditional trap instruction. The system supports a head thread that executes program instructions and a speculative thread that speculatively executes program instructions in advance of the head thread. During operation, the system uses the speculative thread to execute code, which includes an instruction that can cause an exception condition. After the instruction is executed, the system determines if the instruction caused the exception condition. If so, the system writes an exception condition indicator to a register. At some time in the future, the system executes a conditional trap instruction which examines a value in the register. If the value in the register is an exception condition indicator, the system executes a trap handling routine to handle the exception condition. Otherwise, the system proceeds with execution of the code. In one embodiment of the present invention, prior to executing the instruction, the system allows a compiler to optimize a program containing the instruction. This optimization process includes scheduling an exception testing instruction associated with the instruction to occupy a free instruction slot following the instruction. This exception testing instruction determines if the instruction causes the exception condition. In one embodiment of the present invention, the trap handling routine triggers a rollback operation to undo operations performed by the speculative thread.

24 Claims, 14 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 7

☐ 11. Document ID: US 6684398 B2

L4: Entry 11 of 58 File: USPT Jan 27, 2004

US-PAT-NO: 6684398
DOCUMENT-IDENTIFIER: US 6684398 B2

TITLE: Monitor entry and exit for a speculative thread during space and time dimensional execution

DATE-ISSUED: January 27, 2004

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Chaudhry; Shailender	San Francisco	CA		
Tremblay; Marc	Menlo Park	CA		

US-CL-CURRENT: 718/106; 711/145, 711/146, 711/6, 718/107, 719/315, 719/316

ABSTRACT:

One embodiment of the present invention provides a system that facilitates entering and exiting a critical section of code for a speculative thread. The system supports a head thread that executes program instructions, and the speculative thread that speculatively executes program instructions in advance of the head thread. During an entry into the critical section by the speculative thread, the system increments a variable containing a number of virtual locks held by the speculative thread. Note that a virtual lock held by the speculative thread is associated with the critical section and is used to keep track of the fact that the speculative thread has entered the critical section. Also note that this virtual lock does not prevent the speculative thread or other threads from entering the critical section. During an exit from the critical section by the speculative thread, the system decrements the variable containing the number of virtual locks held by the speculative thread. The speculative eventually receives a request to perform a join operation with the head thread to merge state associated with the speculative thread into state associated with the head thread. Upon receiving this request, the speculative thread waits to perform the join operation until the variable containing the number of virtual locks held by the speculative thread equals zero. In one embodiment of the present invention, the system additionally waits to perform the join operation until no virtual locks in a list of virtual locks accessed by the speculative thread are held by the other head threads.

22 Claims, 16 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 7

☐ 12. Document ID: US 6668325 B1

L4: Entry 12 of 58 File: USPT Dec 23, 2003

US-PAT-NO: 6668325
DOCUMENT-IDENTIFIER: US 6668325 B1
**** See image for Certificate of Correction ****

TITLE: Obfuscation techniques for enhancing software security

DATE-ISSUED: December 23, 2003

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Collberg; Christian Sven	Auckland			NZ
Thomborson; Clark David	Auckland			NZ
Low; Douglas Wai Kok	Auckland			NZ

US-CL-CURRENT: 713/194; 713/200

ABSTRACT:

The present invention provides obfuscation techniques for enhancing software security. In one embodiment, a method for obfuscation techniques for enhancing software security includes selecting a subset of code (e.g., compiled source code of an application) to obfuscate, and obfuscating the selected subset of the code. The obfuscating includes applying an obfuscating transformation to the selected subset of the code. The transformed code can be weakly equivalent to the untransformed code. The applied transformation can be selected based on a desired level of security (e.g., resistance to reverse engineering). The applied transformation can include a control transformation that can be creating using opaque constructs, which can be constructed using aliasing and concurrency techniques. Accordingly, the code can be obfuscated for enhanced software security based on a desired level of obfuscation (e.g., based on a desired potency, resilience, and cost).

171 Claims, 55 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 27

☐ 13. Document ID: US 6658451 B1

L4: Entry 13 of 58 File: USPT Dec 2, 2003

US-PAT-NO: 6658451
DOCUMENT-IDENTIFIER: US 6658451 B1

TITLE: Parallel join operation to support space and time dimensional program execution

DATE-ISSUED: December 2, 2003

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Chaudhry; Shailender	San Francisco	CA		
Tremblay; Marc	Menlo Park	CA		

US-CL-CURRENT: 718/108; 709/228

ABSTRACT:

One embodiment of the present invention provides a system that supports space and time dimensional program execution by performing a parallel join operation to merge state created during speculative execution into the non-speculative state of a program. The system executes a program using a head thread that operates on primary versions of memory elements and accesses a primary version of a stack. The system also executes the program using a speculative thread that speculatively executes program instructions in advance of the head thread while the head thread is executing. This speculative thread operates on space-time dimensioned versions of the memory elements and accesses a speculative version of the stack. The system

performs a join operation between the head thread and the speculative thread when the head thread reaches a point in the program where the speculative thread began executing. This join operation involves using both the head thread and the speculative thread to perform a number of operations in parallel. These operations include merging the space-time dimensioned versions of the memory elements into the primary versions of the memory elements so that updates to the space-time dimensioned versions of the memory elements are incorporated into corresponding primary versions of memory elements. These operations also include merging the speculative version of the stack into the primary version of the stack.

25 Claims, 15 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 7

☐ 14. Document ID: US 6634023 B1

L4: Entry 14 of 58 File: USPT Oct 14, 2003

US-PAT-NO: 6634023
DOCUMENT-IDENTIFIER: US 6634023 B1

TITLE: Compile method, exception handling method and computer

DATE-ISSUED: October 14, 2003

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Komatsu; Hideaki	Kanagawa-ken			JP
Oqasawara; Takeshi	Tokyo-to			JP

US-CL-CURRENT: 717/159; 717/124, 717/131, 717/132, 717/133, 717/151, 717/152, 717/154, 717/155, 717/156, 717/157, 717/161

ABSTRACT:

The present invention enables re-ordering of instructions to be executed while assuring a precise exception. In Java language, an optimization process of re-ordering instructions to be executed is performed by Just-In-Time compiler. For instance, the instructions lining in order from instruction E1 which was moved forward to instruction S2which had been located before E1 is registered as interrupt inhibited section R1, and from instruction S4 which was moved forward to instruction S3 which had been located before S4 is registered as interrupt inhibited section R2 (S is an instruction which has an affect observable from the outside at the execution, and E is an instruction which may cause an exception). Also, in FIG. 7, S4 which was an instruction behind E1 in the original order is registered as R1's instruction invalid at an exception. If E1 causes an exception, an interrupt handler is activated and the instructions of interrupt inhibited section R1 are copied to another area. S4 is not copied in that case. In addition, a branch code to an exception handling routine is attached to the end of the copy. If execution is restarted from S1, the instructions required to be executed for assuring the precise exception are executed, and it may move on to an exception handling routine thereafter.

23 Claims, 10 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 7

Full | Title | Citation | Front | Review | Classification | Date | Reference | Claims | KWMC | Draw. D

☐ 15. Document ID: US 6633878 B1

L4: Entry 15 of 58 File: USPT Oct 14, 2003

US-PAT-NO: 6633878
DOCUMENT-IDENTIFIER: US 6633878 B1

TITLE: Initializing an ecommerce database framework

DATE-ISSUED: October 14, 2003

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Underwood; Roy Aaron	Long Grove	IL		

US-CL-CURRENT: 707/100; 707/1, 707/102, 707/205

ABSTRACT:

A system, method and article of manufacture are provided for initializing a database used with an issue tracker. The issue tracker receives information relating to a plurality of issues from a plurality of users, displays the information relating to the issues, and allows the browsing of the information relating to each of the issues. To initialize the database, the information relating to the issues is stored in a first database. A second database is also provided that stores tables including: a plurality of user interfaces; and/or application logic for accessing the information in the first database. The tables of the second database are reconfigured upon migrating the first database from a first folder to a second folder.

15 Claims, 179 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 111

Full | Title | Citation | Front | Review | Classification | Date | Reference | Claims | KWMC | Draw. D

☐ 16. Document ID: US 6625751 B1

L4: Entry 16 of 58 File: USPT Sep 23, 2003

US-PAT-NO: 6625751
DOCUMENT-IDENTIFIER: US 6625751 B1

TITLE: Software fault tolerant computer system

DATE-ISSUED: September 23, 2003

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Starovic; Gradimir	Pinner			GB
Mayhead; Martin P	Hindhead			GB

US-CL-CURRENT: 714/11; 714/13

ABSTRACT:

A software fault tolerant computer system includes a primary virtual machine and a secondary virtual machine. The secondary virtual machine is operable to replicate the primary virtual machine by replicating operations performed on the primary virtual machine. The primary and the secondary virtual machines are further operable to test for equivalent operation of the primary and secondary virtual machines at predetermined stages of operation. This provides software fault tolerance wherein both a unit of replication and a component that implements the fault tolerance mechanisms is a virtual machine (VM). Since a VM as used by the invention has full knowledge of the semantics of application-level code, fault tolerance mechanisms can be provided by the VMs without requiring any increase in application complexity. Co-ordination of replicated states and computations is achieved with characteristics of both active and passive replication.

38 Claims, 22 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 9

☐ 17. Document ID: US 6615166 B1

L4: Entry 17 of 58 File: USPT Sep 2, 2003

US-PAT-NO: 6615166
DOCUMENT-IDENTIFIER: US 6615166 B1

TITLE: Prioritizing components of a network framework required for implementation of technology

DATE-ISSUED: September 2, 2003

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Guheen; Michael F.	Tiburon	CA		
Mitchell; James D.	Manhattan Beach	CA		
Barrese; James J.	San Jose	CA		

US-CL-CURRENT: 703/27; 703/26, 709/220, 709/223, 709/231, 717/140, 719/316

ABSTRACT:

A system and method are provided for prioritizing components of an existing network framework. First, a plurality of components required for implementation of a predetermined technology using an existing network framework are provided. Next, a priority listing of the components is complied such that the relative position of the components on the priority listing corresponds to a temporal priority among the components. The existing network framework and the components are pictorially represented. Next, a first component of the existing network framework is indicia coded in order to indicate that the first component must be installed first based on the component's position on the priority listing. Thereafter, a second component and any remaining components of the existing network framework is indicia encoded in order to indicate that the second component and any remaining components must be installed after the first component based on the second component's position on the priority listing.

18 Claims, 177 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 177

☐ 18. Document ID: US 6609128 B1

L4: Entry 18 of 58 File: USPT Aug 19, 2003

US-PAT-NO: 6609128
DOCUMENT-IDENTIFIER: US 6609128 B1

TITLE: Codes table framework design in an E-commerce architecture

DATE-ISSUED: August 19, 2003

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Underwood; Roy Aaron	Long Grove	IL		

US-CL-CURRENT: 707/10; 707/200

ABSTRACT:

A system, method and article of manufacture are provided for maintaining application consistency. First, a table of codes and associated text phrases are provided. Such table of codes is stored on a local storage medium within an e-commerce computer architecture. Next, the table of codes is accessed on the local storage medium within the e-commerce computer architecture. One of the text phrases is subsequently retrieved by selecting a corresponding one of the codes of the table. During operation, modification of the text phrases associated with each of the codes of the table is permitted. A plurality of services are executed, including retrieving a single one of the text phrases, retrieving all of the text phrases in response to a single command, updating a single code and text phrase combination, updating all of the code and text phrase combinations, naming the table, adding a new code and text phrase combination, removing one of the code and text phrase combination, and adding another table.

15 Claims, 179 Drawing figures

Exemplary Claim Number: 1
Number of Drawing Sheets: 111

Full | Title | Citation | Front | Review | Classification | Date | Reference | Claims | KWMC | Draw. D

☐ 19. Document ID: US 6601233 B1

L4: Entry 19 of 58 File: USPT Jul 29, 2003

US-PAT-NO: 6601233
DOCUMENT-IDENTIFIER: US 6601233 B1

TITLE: Business components framework

DATE-ISSUED: July 29, 2003

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Underwood; Roy Aaron	Long Grove	IL		

US-CL-CURRENT: 717/102; 717/100, 717/101, 717/103, 717/104, 717/106, 717/107

ABSTRACT:

A method of generating software based on business components. A plurality of logical business components in a business are first defined with each business component having a plurality of capabilities. Next, functional interrelationships are identified between the logical business components. Code modules are then generated to carry out the capabilities of the logical business components and the functional interrelationships between the logical business components, wherein the code modules represent a transformation of the logical business components to their physical implementation, while ensuring the capabilities that are carried out by each code module are essentially unique to the logical business component associated with the code module. Next, the functional aspects of the code modules and the functional relationships of the code modules are tested. The code modules are then subsequently deployed in an e-commerce environment.

18 Claims, 177 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 111

Full | Title | Citation | Front | Review | Classification | Date | Reference | Claims | KWMC | Draw. D

☐ 20. Document ID: US 6536037 B1

L4: Entry 20 of 58 File: USPT Mar 18, 2003

US-PAT-NO: 6536037
DOCUMENT-IDENTIFIER: US 6536037 B1
** See image for Certificate of Correction **

TITLE: Identification of redundancies and omissions among components of a web based architecture

DATE-ISSUED: March 18, 2003

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Guheen; Michael F	Tiburon	CA		
Mitchell; James D.	Manhattan Beach	CA		
Barrese; James J.	San Jose	CA		

US-CL-CURRENT: 717/151; 703/2, 709/231

ABSTRACT:

A system, method and article of manufacture are provided for conveying redundancies and omissions among components of a network framework such as a web architecture framework. First, an area of an existing network framework is determined in which redundancies and omissions exist. Next, a pictorial representation of the existing network framework is presented along with a plurality of its components. The foregoing redundancies and the omissions are then highlighted by indicia coding the components of the existing network that reside in the area. As such, a diagnostic analysis of redundant efforts and gaps in a current implementation of the existing network framework is effectively conveyed.

19 Claims, 177 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 177

☐ 21. Document ID: US 6526570 B1

L4: Entry 21 of 58 File: USPT Feb 25, 2003

US-PAT-NO: 6526570
DOCUMENT-IDENTIFIER: US 6526570 B1
**** See image for Certificate of Correction ****

TITLE: File portability techniques

DATE-ISSUED: February 25, 2003

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Click, Jr.; Clifford N.	San Jose	CA		
Vick; Christopher A.	San Jose	CA		
Paleczny; Michael H.	Sunnyvale	CA		

US-CL-CURRENT: 717/146; 717/140, 717/147

ABSTRACT:

Systems and methods for building a platform specific compiler in a multi-platform environment are provided. A set of user defined platform dependent compiler architecture descriptors that describe corresponding architectural features of a particular hardware platform dependent compiler are provided. The descriptors are converted into the platform dependent compiler source code which is compiled into platform dependent compiler object code. The platform specific compiler is formed from the platform dependent compiler object code and platform independent compiler object code already provide. During compiler run time an interface mediates the flow of information between the platform dependent compiler object code and the platform independent compiler object code.

16 Claims, 10 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 10

☐ 22. Document ID: US 6526421 B1

L4: Entry 22 of 58 File: USPT Feb 25, 2003

US-PAT-NO: 6526421
DOCUMENT-IDENTIFIER: US 6526421 B1

TITLE: Method of scheduling garbage collection

DATE-ISSUED: February 25, 2003

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Houldsworth; Richard J.	Horley			GB

US-CL-CURRENT: 707/206; 707/101

ABSTRACT:

A method of scheduling instructions to be executed concurrently by a processor, the processor being capable of executing a predetermined number of instructions concurrently. Instructions from a first process and a second process are interleaved according to a predetermined rule to give a third process. Instructions from the third process are then scheduled for execution at a first time point by the processor. Instructions of the first process generate data structures comprising data objects linked by identifying pointers in a memory heap. The second process comprises a garbage collection process for traversing the memory heap and reclaiming memory allocated to data structures unused by the first process.

11 Claims, 5 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 3

☐ 23. Document ID: US 6523027 B1

L4: Entry 23 of 58 File: USPT Feb 18, 2003

US-PAT-NO: 6523027
DOCUMENT-IDENTIFIER: US 6523027 B1

TITLE: Interfacing servers in a Java based e-commerce architecture

DATE-ISSUED: February 18, 2003

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Underwood; Roy Aaron	Long Grove	IL		

US-CL-CURRENT: 707/4; 707/10, 707/100

ABSTRACT:

A system, method and article of manufacture are provided for providing an interface between a first server and a second server with a proxy component situated therebetween. Initially, a request for a business object is identified by an application on the first server. The first server is then connected to the second server. Next, selection criteria from the first server is transmitted to the second server. In response to the selection criteria, the first server receives a first recordset and a second recordset from the second server. Business data is included in the first recordset and result codes are included in the second recordset. The first and second recordsets are mapped to the business object and the business object is sent to the application on the first server.

18 Claims, 179 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 111

Full | Title | Citation | Front | Review | Classification | Date | Reference | | | Claims | KWIC | Draw. D

☐ 24. Document ID: US 6519571 B1

L4: Entry 24 of 58 File: USPT Feb 11, 2003

US-PAT-NO: 6519571
DOCUMENT-IDENTIFIER: US 6519571 B1

TITLE: Dynamic customer profile management

DATE-ISSUED: February 11, 2003

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Guheen; Michael F.	Tiburon	CA		

Mitchell; James D.	Manhatten Beach	CA
Barrese; James J.	San Jose	CA

US-CL-CURRENT: 705/14

ABSTRACT:

The present invention is provided for utilizing various types of user indicia such as search requests, products purchased, products looked at but not purchased, products purchased and returned, reasons for returning products, customers stated profile including income level, education level, stated profession, etc. for the purpose of customizing a user interface.

17 Claims, 177 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 177

Full | Title | Citation | Front | Review | Classification | Date | Reference | Claims | KWIC | Draw De

☐ 25. Document ID: US 6507946 B2

L4: Entry 25 of 58 File: USPT Jan 14, 2003

US-PAT-NO: 6507946
DOCUMENT-IDENTIFIER: US 6507946 B2

TITLE: Process and system for Java virtual method invocation

DATE-ISSUED: January 14, 2003

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Alexander, III; William Preston	Austin	TX		
Gu; Weiming	Austin	TX		

US-CL-CURRENT: 717/145; 714/35, 717/140, 717/151, 717/157, 717/158, 719/315

ABSTRACT:

A process and system for optimizing an invocation of a method is provided. A determination is made to compile a calling method, and a call to a callee method is detected within the first method. The callee method may be a non-final, virtual method, and a determination may be made that the callee method has not been previously overridden. The callee method is then inlined within the first method. In addition, no conditional statements are inserted into the calling method along with the inlined method. The determination to compile and optimize these methods may be made by a just-in-time compiler, and if the methods are Java methods, then a Java just-in-time compiler performs the optimization process. If a determination is made to load a class that contains a method that overrides the callee method, then the calling method is recompiled or patched.

17 Claims, 18 Drawing figures
Exemplary Claim Number: 1

Number of Drawing Sheets: 5

Full | Title | Citation | Front | Review | Classification | Date | Reference | Claims | KWMC | Draw De

☐ 26. Document ID: US 6473794 B1

L4: Entry 26 of 58 File: USPT Oct 29, 2002

US-PAT-NO: 6473794
DOCUMENT-IDENTIFIER: US 6473794 B1

TITLE: System for establishing plan to test components of web based framework by displaying pictorial representation and conveying indicia coded components of existing network framework

DATE-ISSUED: October 29, 2002

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Guheen; Michael F.	Tiburon	CA		
Mitchell; James D.	Manhattan Beach	CA		
Barrese; James J.	San Jose	CA		

US-CL-CURRENT: 709/223; 709/224

ABSTRACT:

A system, method, and article of manufacture are provided for planning the testing of components of an existing network framework. First, a pictorial representation of an existing network framework is displayed along with a plurality of components of the existing network framework. Thereafter, the components of the existing network framework are indicia coded in order to convey a plan by which the components of the existing network framework are to be tested. The components may be indicia coded in order to convey an order of the testing or which components of the existing network framework are to be tested.

19 Claims, 177 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 177

Full | Title | Citation | Front | Review | Classification | Date | Reference | Claims | KWMC | Draw De

☐ 27. Document ID: US 6463582 B1

L4: Entry 27 of 58 File: USPT Oct 8, 2002

US-PAT-NO: 6463582
DOCUMENT-IDENTIFIER: US 6463582 B1

TITLE: Dynamic optimizing object code translator for architecture emulation and

dynamic optimizing object code translation method

DATE-ISSUED: October 8, 2002

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Lethin; Richard A.	New York	NY		
Bank, III; Joseph A.	New York	NY		
Garrett; Charles D.	Seattle	WA		
Wada; Mikayo	Kawasaki			JP
Sakurai; Mitsuo	Kawasaki			JP

US-CL-CURRENT: 717/158; 717/138, 717/139

ABSTRACT:

An optimizing object code translation system and method perform dynamic compilation and translation of a target object code on a source operating system while performing optimization. Compilation and optimization of the target code is dynamically executed in real time. A compiler performs analysis and optimizations that improve emulation relative to template-based translation and interpretation such that a host processor which processes larger order instructions, such as 32-bit instructions, may emulate a target processor which processes smaller order instructions, such as 16-bit and 8-bit instructions. The optimizing object code translator does not require knowledge of a static program flow graph or memory locations of target instructions prior to run time. In addition, the optimizing object code translator does not require knowledge of the location of all join points into the target object code prior to execution. During program execution, a translator records branch operations. The logging of information identifies instructions and instruction join points. When a number of times a branch operation is executed exceeds a threshold, the destination of the branch becomes a seed for compilation and code portions between seeds are defined as segments. A segment may be incomplete allowing for modification or replacement to account for a new flow of program control during real time program execution.

32 Claims, 37 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 19

☐ 28. Document ID: US 6463526 B1

L4: Entry 28 of 58 File: USPT Oct 8, 2002

US-PAT-NO: 6463526
DOCUMENT-IDENTIFIER: US 6463526 B1

TITLE: Supporting multi-dimensional space-time computing through object versioning

DATE-ISSUED: October 8, 2002

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Chaudhry; Shailender	San Francisco	CA		
Tremblay; Marc	Menlo Park	CA		

US-CL-CURRENT: 712/228; 718/107

ABSTRACT:

One embodiment of the present invention provides a system that facilitates multi-dimensional space and time dimensional execution of computer programs. The system includes a head thread that executes program instructions and a series of speculative threads that execute program instructions in advance of the head thread, wherein each speculative thread executes program instructions in advance of preceding speculative threads in the series. The head thread accesses a primary version of the memory element and the series of speculative threads access space-time dimensioned versions of the memory element. The system starts by receiving a memory access to the memory element. If the memory access is a write operation by the head thread or a speculative thread, the system determines if a version of the memory element associated with the head thread or speculative thread exists. If not, the system creates a version of the memory element for the thread. Next, the system performs the write operation to the version of the memory element. After performing the write operation, the system checks status information associated with the memory element to determine if the memory element has been read by a following speculative thread in the series of speculative threads. If so, the system causes the following speculative thread and any successive speculative threads in the series to roll back so that the following speculative thread and any successive speculative threads in the series can read a result of the write operation. If not, the system performs the write operation to all successive space-time dimensioned versions of the memory element.

20 Claims, 25 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 11

☐ 29. Document ID: US 6460067 B1

L4: Entry 29 of 58 File: USPT Oct 1, 2002

US-PAT-NO: 6460067
DOCUMENT-IDENTIFIER: US 6460067 B1

TITLE: Using time stamps to improve efficiency in marking fields within objects

DATE-ISSUED: October 1, 2002

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Chaudhry; Shailender	San Francisco	CA		
Tremblay; Marc	Menlo Park	CA		

US-CL-CURRENT: 718/101; 718/102, 718/103, 718/104

ABSTRACT:

One embodiment provides for a system that uses a time stamp in order to more efficiently mark objects to keep track of accesses to fields with the objects. Upon receiving a first reference to a first field in an object, the system determines whether the first field has been accessed within a current time period. The system does so by retrieving a time stamp associated with the object. This time stamp indicates the last time any marking bit associated with any field in the object was updated. The system compares the time stamp with a current time value associated with the current time period. The system additionally retrieves a first marking bit associated with the first field and determines if the first marking bit is set. If the first marking bit is set and if the time stamp equals the current time value, the system determines that the first field has been accessed in the current time period. The system indicates that a second field in the object has been accessed in the current time period upon receiving a second reference to the second field. In response to the second reference, the system sets a second marking bit associated with the second field. The system also updates the time stamp associated with the object, if necessary, so that the time stamp contains the current time value.

14 Claims, 17 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 9

☐ 30. Document ID: US 6453463 B1

L4: Entry 30 of 58 File: USPT Sep 17, 2002

US-PAT-NO: 6453463
DOCUMENT-IDENTIFIER: US 6453463 B1
**** See image for Certificate of Correction ****

TITLE: Method and apparatus for providing finer marking granularity for fields within objects

DATE-ISSUED: September 17, 2002

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Chaudhry; Shailender	San Francisco	CA		
Tremblay; Marc	Menlo Park	CA		

US-CL-CURRENT: 717/127

ABSTRACT:

One embodiment of the present invention provides a system that facilitates marking of objects defined within an object-oriented programming system to keep track of accesses to fields within the objects. The system operates by receiving a reference to a field within an object, and identifying a marking bit within the object that is associated with the field. Note that each marking bit within the object is associated with a different subset of fields within the object. Next, the system sets the marking bit to indicate that at least one field within the associated

subset of fields has been referenced. Finally, the system performs the reference to the field. In one embodiment of the present invention, the object includes N marking bits numbered 0, 1, 2, . . . , N-1 and M fields numbered 0, 1, 2, . . . , M-1. In this embodiment, the system identifies the marking bit associated with the field by starting with a field number for the field, and applying a modulo N operation to the field number to produce a number for the associated marking bit. In a variation on this embodiment, N is a power of two.

22 Claims, 15 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 7

☐ 31. Document ID: US 6438677 B1

L4: Entry 31 of 58 File: USPT Aug 20, 2002

US-PAT-NO: 6438677
DOCUMENT-IDENTIFIER: US 6438677 B1

TITLE: Dynamic handling of object versions to support space and time dimensional program execution

DATE-ISSUED: August 20, 2002

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Chaudhry; Shailender	San Francisco	CA		
Tremblay; Marc	Menlo Park	CA		

US-CL-CURRENT: 712/27; 712/202, 712/228, 712/23, 717/108, 717/118, 717/165, 718/106, 718/108, 719/315, 719/316, 719/332

ABSTRACT:

One embodiment of the present invention provides a system that supports space and time dimensional program execution by facilitating accesses to different versions of a memory element. The system supports a head thread that executes program instructions and a speculative thread that executes program instructions in advance of the head thread. The head thread accesses a primary version of the memory element, and the speculative thread accesses a space-time dimensioned version of the memory element. During a reference to the memory element by the head thread, the system accesses the primary version of the memory element. During a reference to the memory element by the speculative thread, the speculative thread accesses a pointer associated with the primary version of the memory element, and accesses a version of the memory element through the pointer. Note that the pointer points to the space-time dimensioned version of the memory element if the space-time dimensioned version of the memory element exists. In one embodiment of the present invention, the pointer points to the primary version of the memory element if the space-time dimensioned version of the memory element does not exist.

24 Claims, 16 Drawing figures
Exemplary Claim Number: 1

Number of Drawing Sheets: 7

Full | Title | Citation | Front | Review | Classification | Date | Reference | Claims | KWMC | Draw. De

☐ 32. Document ID: US 6430649 B1

L4: Entry 32 of 58 File: USPT Aug 6, 2002

US-PAT-NO: 6430649
DOCUMENT-IDENTIFIER: US 6430649 B1

TITLE: Method and apparatus for enforcing memory reference dependencies through a load store unit

DATE-ISSUED: August 6, 2002

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Chaudhry; Shailender	San Francisco	CA		
Tremblay; Marc	Menlo Park	CA		
O'Connor; James M.	Union City	CA		

US-CL-CURRENT: 711/100; 712/216, 712/225

ABSTRACT:

One embodiment of the present invention provides a system that enforces dependencies between memory references within a load store unit (LSU) in a processor. When a write request is received in the load store unit, the write request is loaded into a store buffer in the LSU. The write request may include a "watch address" specifying that a subsequent load from the watch address cannot occur before the write request completes. Note that the watch address is not necessarily the same as the destination address of the write operation. When a read request is received in the load store unit, the read request is loaded into a load buffer of the LSU. The system determines if the read request is directed to the same address as a matching watch address in the store buffer. If so, the system waits for the write request associated with the matching watch address to complete before completing the read request. In one embodiment of the present invention, if the read request is directed to the same address as a matching write request in the store buffer, the system completes the read request by returning a data value contained in the matching write request without going out to memory. In one embodiment of the present invention, the system provides an executable code write instruction that specifies the watch address.

25 Claims, 16 Drawing figures
Exemplary Claim Number: 24,25
Number of Drawing Sheets: 9

Full | Title | Citation | Front | Review | Classification | Date | Reference | Claims | KWMC | Draw. De

☐ 33. Document ID: US 6404738 B1

L4: Entry 33 of 58 File: USPT Jun 11, 2002

US-PAT-NO: 6404738
DOCUMENT-IDENTIFIER: US 6404738 B1

TITLE: Dynamic network bandwidth allocation for multimedia applications with soft quality-of-service requirements

DATE-ISSUED: June 11, 2002

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Reininger; Daniel	Princeton	NJ		
Izmailov; Rauf	Middletown	NJ		

US-CL-CURRENT: 370/236; 370/465

ABSTRACT:

A new concept of soft quality-of-service (soft-QoS) is developed that bridges the gap between the efficient provision of network-level QoS and the requirements of multimedia applications. Soft-QoS is defined by a satisfaction index (a number that rates users' perceptual quality) and a softness profile (a function that captures the robustness of multimedia applications to network congestion). Another aspect of this invention is a bandwidth allocation scheme for multimedia applications with soft-QoS requirements is presented. The implementation of the bandwidth allocation scheme on a network element realizes a soft-QoS controller. The controller uses the connections' softness profiles to compute a bandwidth allocation that maximizes the minimum satisfaction index of active connections.

20 Claims, 25 Drawing figures
Exemplary Claim Number: 17
Number of Drawing Sheets: 14

☐ 34. Document ID: US 6374351 B1

L4: Entry 34 of 58 File: USPT Apr 16, 2002

US-PAT-NO: 6374351
DOCUMENT-IDENTIFIER: US 6374351 B1
**** See image for Certificate of Correction ****

TITLE: Software branch prediction filtering for a microprocessor

DATE-ISSUED: April 16, 2002

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY

Tremblay; Marc Menlo Park CA

US-CL-CURRENT: 712/239

ABSTRACT:

The present invention provides software branch prediction filtering for a microprocessor. In one embodiment, a method for a software branch prediction filtering for a microprocessor includes determining whether a branch is "easy" to predict, and predicting the branch using software branch prediction if the branch is easy to predict. Otherwise (i.e., the branch is "hard" to predict), the branch is predicted using hardware branch prediction. Accordingly, more accurate but space-limited hardware branch prediction resources are conserved for hard-to-predict branches.

36 Claims, 5 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 4

☐ 35. Document ID: US 6366876 B1

L4: Entry 35 of 58 File: USPT Apr 2, 2002

US-PAT-NO: 6366876
DOCUMENT-IDENTIFIER: US 6366876 B1

TITLE: Method and apparatus for assessing compatibility between platforms and applications

DATE-ISSUED: April 2, 2002

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Looney; Kevin T.	Sunnyvale	CA		

US-CL-CURRENT: 703/25; 703/26, 707/10, 713/1

ABSTRACT:

Embodiments of the invention can be used to assess whether a software application is compatible with an operating platform. A specification that describes the operating platform is generated using a definitional language. The specification identifies the programming resources of an operating platform. The application's dependencies and programming resources are identified. A compatibility engine is executed to resolve an application's dependencies to the specification. The output of the compatibility engine identifies whether the application conforms to the operating platform and how it deviates from the specification.

37 Claims, 14 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 14

☐ 36. Document ID: US 6353923 B1

L4: Entry 36 of 58 File: USPT Mar 5, 2002

US-PAT-NO: 6353923
DOCUMENT-IDENTIFIER: US 6353923 B1

TITLE: Active debugging environment for debugging mixed-language scripting code

DATE-ISSUED: March 5, 2002

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Bogle; Phillip Lee	Redmond	WA		
Katzenberger; Gary S.	Woodinville	WA		
McKelvie; Samuel James	Seattle	WA		
Welland; Robert Victor	Seattle	WA		

US-CL-CURRENT: 717/128

ABSTRACT:

An active debugging environment for debugging a virtual application that contains program language code from multiple compiled and/or interpreted programming languages. The active debugging environment is language neutral and host neutral, where the host is a standard content centric script host with language engines for each of the multiple compiled and/or interpreted programming languages represented in the virtual application. The active debugging environment user interface can be of any debug tool interface design. The language neutral and host neutral active debugging environment is facilitated by a process debug manager that catalogs and manages application specific components, and a machine debug manager that catalogs and manages the various applications that comprise a virtual application being run by the script host. The process debug manager and the machine debug manager act as an interface between the language engine specific programming language details and the debug user interface.

20 Claims, 7 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 6

Full | Title | Citation | Front | Review | Classification | Date | Reference | Claims | KWMC | Draw De

☐ 37. Document ID: US 6353881 B1

L4: Entry 37 of 58 File: USPT Mar 5, 2002

US-PAT-NO: 6353881

DOCUMENT-IDENTIFIER: US 6353881 B1

TITLE: Supporting space-time dimensional program execution by selectively versioning memory updates

DATE-ISSUED: March 5, 2002

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Chaudhry; Shailender	Santa Clara	CA		
Tremblay; Marc	Menlo Park	CA		

US-CL-CURRENT: 712/228; 711/168, 711/169, 712/206, 712/215, 712/219, 712/23, 712/27, 718/107, 718/108, 719/315, 719/316 , 719/332

ABSTRACT:

A system is provided that facilitates space and time dimensional execution of computer programs through selective versioning of memory elements located in a system heap. The system includes a head thread that executes program instructions and a speculative thread that simultaneously executes program instructions in advance of the head thread with respect to the time dimension of sequential execution of the program. The collapsing of the time dimensions is facilitated by expanding the heap into two space-time dimensions, a primary dimension (dimension zero), in which the head thread operates, and a space-time dimension (dimension one), in which the speculative thread operates. In general, each dimension contains its own version of an object and objects created by the thread operating in the dimension. The head thread generally accesses a primary version of a memory element and the speculative thread generally accesses a corresponding space-time dimensioned version of the memory element. During a write by the head thread, the system performs the write to all dimensions of the memory element. Note that if the dimensions are collapsed at this address a single update will update all time dimensions. It also checks status information associated with the memory element to determine if the memory element has been read by the speculative thread. If so, the system causes the speculative thread to roll back so that the speculative thread can read a result of the write operation.

23 Claims, 12 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 5



☐ 38. Document ID: US 6341348 B1

L4: Entry 38 of 58 File: USPT Jan 22, 2002

US-PAT-NO: 6341348
DOCUMENT-IDENTIFIER: US 6341348 B1

TITLE: Software branch prediction filtering for a microprocessor

DATE-ISSUED: January 22, 2002

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Tremblay; Marc	Menlo Park	CA		

US-CL-CURRENT: 712/239

ABSTRACT:

The present invention provides software branch prediction filtering for a microprocessor. In one embodiment, a method for a software branch prediction filtering for a microprocessor includes determining whether a branch is "easy" to predict, and predicting the branch using software branch prediction if the branch is easy to predict. Otherwise (i.e., the branch is "hard" to predict), the branch is predicted using hardware branch prediction. Accordingly, more accurate but space-limited hardware branch prediction resources are conserved for hard-to-predict branches.

16 Claims, 5 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 4

☐ 39. Document ID: US 6324686 B1

L4: Entry 39 of 58 File: USPT Nov 27, 2001

US-PAT-NO: 6324686
DOCUMENT-IDENTIFIER: US 6324686 B1

TITLE: Just in time compiler technique

DATE-ISSUED: November 27, 2001

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Komatsu; Hideaki	Yokohama			JP
Momose; Hiroyuki	Tokyo-to			JP
Ogata; Kazunori	Fujisawa			JP

US-CL-CURRENT: 717/148; 717/118, 717/146, 717/154

ABSTRACT:

The present invention provides a compiler that can compile Java byte code at high speed while employing only limited resources. The technique converts byte code into intermediate code that employs, as an opcode, an address of an execution routine that corresponds to a byte code, and executes the execution routines in a sequential order corresponding to that of the intermediate code. The length of the intermediate code is constant and the bit location of the opcode is also constant. The execution routine is stored in sequential areas in a memory, and an offset of the execution routine group from the head address of the sequential memory area is employed as an opcode. Furthermore, by employing empty pipelines in the CPU, one

part (operand generation and a jump) of an instruction group for jumping to the next intermediate code execution routine, and one part (prefetching of intermediate code and address calculation of a execution routine) of an instruction group for jumping to the next plus one intermediate execution routine can be executed in parallel to the original intermediate code process. Further, rule-based optimization using idioms is performed, without requiring analysis of the program structure.

18 Claims, 20 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 15

☐ 40. Document ID: US 6286134 B1

L4: Entry 40 of 58 File: USPT Sep 4, 2001

US-PAT-NO: 6286134
DOCUMENT-IDENTIFIER: US 6286134 B1
**** See image for Certificate of Correction ****

TITLE: Instruction selection in a multi-platform environment

DATE-ISSUED: September 4, 2001

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Click, Jr.; Clifford N.	San Jose	CA		
Vick; Christopher A.	San Jose	CA		
Paleczny; Michael H.	Sunnyvale	CA		

US-CL-CURRENT: 717/138; 717/141

ABSTRACT:

Systems and methods for building a platform specific compiler having an embedded instruction selector in a multi-platform environment are provided. A set of user defined platform dependent compiler architecture descriptors that describe corresponding architectural features and a set of instruction predicates defining those instructions to be selected are converted to platform dependent compiler object code and instruction selector object code, respectively. The platform specific compiler having the embedded instruction selector is formed from the platform dependent compiler object code, the instruction selector object code, and the platform independent compiler object code.

16 Claims, 12 Drawing figures
Exemplary Claim Number: 7
Number of Drawing Sheets: 12

☐ 41. Document ID: US 6272674 B1

L4: Entry 41 of 58 File: USPT Aug 7, 2001

US-PAT-NO: 6272674
DOCUMENT-IDENTIFIER: US 6272674 B1

TITLE: Method and apparatus for loading a Java application program

DATE-ISSUED: August 7, 2001

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Holiday, Jr.; Matthew R.	Allen	TX		

US-CL-CURRENT: 717/174

ABSTRACT:

An apparatus and method for loading software into a Java virtual machine ("JVM") in a manner suited for real-time server applications. The software to be loaded is organized by Java package and class so that an application may be loaded in units of packages. Each package, and each class within a package, is loaded into the JVM in an order such that no package or class is loaded before the packages or classes upon which it depends. All software for an application is loaded into the JVM, and any compilation, optimization, or initialization takes place, prior to execution of the application program, so that no delays are incurred during such execution. Software loaded into the JVM, as well as attributes of that software, are identified. Versions of packages are compared when loading the packages to ensure compatibility. An "image" of loaded software is created, which image may be reused by the JVM in order to restart an application rapidly following a failure. A loader environment within the JVM contains information about all loaded applications, packages, and classes, their attributes, and their interrelationships.

27 Claims, 4 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 4

☐ 42. Document ID: US 6256784 B1

L4: Entry 42 of 58 File: USPT Jul 3, 2001

US-PAT-NO: 6256784
DOCUMENT-IDENTIFIER: US 6256784 B1

TITLE: Interpreter with reduced memory access and improved jump-through-register handling

DATE-ISSUED: July 3, 2001

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Grove; Daniel D.	San Jose	CA		

US-CL-CURRENT: 717/139; 711/1, 717/148

ABSTRACT:

The present invention provides an interpreter with reduced memory access and improved jump-through-register handling. In one embodiment, a method includes storing a handler for a bytecode in a cell of a predetermined size of a table, and generating an address of the handler for the bytecode using a shift and an ADD operation. In particular, the handler address is generated by adding a base address of the table and an offset into the table. In another embodiment, a method includes prefetching a target handler address for providing improved jump-through-register handling.

15 Claims, 17 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 10

☐ 43. Document ID: US 6247027 B1

L4: Entry 43 of 58 File: USPT Jun 12, 2001

US-PAT-NO: 6247027
DOCUMENT-IDENTIFIER: US 6247027 B1

TITLE: Facilitating garbage collection during object versioning for space and time dimensional computing

DATE-ISSUED: June 12, 2001

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Chaudhry; Shailender	San Francisco	CA		
Tremblay; Marc	Menlo Park	CA		

US-CL-CURRENT: 707/206; 707/100, 707/203, 717/170

ABSTRACT:

One embodiment of the present invention provides a system that facilitates garbage collection and supports space and time dimensional execution of a computer program. The system executes program instructions with a head thread and speculatively executes program instructions in advance of the head thread with a speculative thread. During execution of the speculative thread, the system creates space-time dimensioned versions of objects from a system heap that are modified by the speculative thread. These space-time dimensioned versions of objects are created in a speculative heap that is separate from the system heap. The system keeps a record of objects for which space-time dimensioned versions have been created during

updates to value fields and during updates to pointer fields by the speculative thread. This record is used during a garbage collection operation to identify live objects so that the garbage collection operation can move the live objects from the speculative heap to the system heap. In one embodiment of the present invention, if the speculative thread causes a hazard condition, the system performs a rollback. This rollback uses the record to identify objects in the system heap that have been modified by the speculative thread so that the modifications can be undone. Note that a hazard condition can occur if the head thread writes to a field that was read by the speculative thread, or alternatively if the head thread writes to a space-time dimensioned version of an object that was written to by the speculative thread.

21 Claims, 16 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 8

☐ 44. Document ID: US 6223346 B1

L4: Entry 44 of 58 File: USPT Apr 24, 2001

US-PAT-NO: 6223346
DOCUMENT-IDENTIFIER: US 6223346 B1

TITLE: Computer program product having preloaded software module

DATE-ISSUED: April 24, 2001

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Tock; Theron D.	Sunnyvale	CA		

US-CL-CURRENT: 717/166; 713/1, 713/2

ABSTRACT:

A method and system for providing an executable module having an address space for storing program data that is to reside in a read-only storage medium and an address space for storing program data that is to reside in a random access memory is herein described. The executable module represents Java classes that are structured for dynamic class loading. A static class loader is used to modify the class structure to accommodate static loading. The static class loader also identifies methods that contain unresolved symbolic references and data that varies during the execution of the module. These methods and data are identified in order to place them in the address space that resides in the random access memory. The static loader is beneficial in a distributed computing environment having a client computer that has little or no secondary storage thereby requiring applications to run entirely in random access memory. By utilizing a read-only memory to store statically loadable classes, the random access memory is left available for other uses.

7 Claims, 14 Drawing figures
Exemplary Claim Number: 1

Number of Drawing Sheets: 12

Full | Title | Citation | Front | Review | Classification | Date | Reference | Claims | KWMC | Draw. D

☐ 45. Document ID: US 6205578 B1

L4: Entry 45 of 58 File: USPT Mar 20, 2001

US-PAT-NO: 6205578
DOCUMENT-IDENTIFIER: US 6205578 B1

TITLE: Interpreter for stack-based languages

DATE-ISSUED: March 20, 2001

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Grove; Daniel D.	San Jose	CA		

US-CL-CURRENT: 717/118; 717/139

ABSTRACT:

The present invention provides an improved interpreter for stack-based languages. In one embodiment, a method includes executing a first interpreter for a first state, and executing a second interpreter for a second state. In particular, the first state indicates that no elements of a stack are stored in registers of a microprocessor, and a second state indicates that an element of the stack is stored in a register of the microprocessor.

21 Claims, 17 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 10

Full | Title | Citation | Front | Review | Classification | Date | Reference | Claims | KWMC | Draw. D

☐ 46. Document ID: US 6151618 A

L4: Entry 46 of 58 File: USPT Nov 21, 2000

US-PAT-NO: 6151618
DOCUMENT-IDENTIFIER: US 6151618 A
**** See image for Certificate of Correction ****

TITLE: Safe general purpose virtual machine computing system

DATE-ISSUED: November 21, 2000

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY

Wahbe; Robert S.	Seattle	WA
Lucco; Steven E.	Seattle	WA
Fraser; Christopher W.	Seattle	WA
Morrison; Vance Palmer	Kirkland	WA

US-CL-CURRENT: 718/1; 717/118, 717/124, 717/148

ABSTRACT:

A safe general purpose virtual machine computing system having a general purpose memory protection model that is hardware architecture and programming language independent. The safe general purpose virtual machine computing system is software based to facilitate operation on hardware architectures that otherwise would prevent the exchange and successful execution of mobile code programs from one computer system to another. The safe general purpose virtual machine computing system also facilitates generating Bytecode Reduced Instruction Set Computer (BRISC) compressed mobile code that can be compiled or translated into executable code very quickly in addition to being compact for transmission purposes, and that is prevented from accessing unauthorized memory locations due to Software Fault Isolation techniques implemented in the code.

38 Claims, 8 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 7

☐ 47. Document ID: US 6148391 A

L4: Entry 47 of 58 File: USPT Nov 14, 2000

US-PAT-NO: 6148391
DOCUMENT-IDENTIFIER: US 6148391 A

TITLE: System for simultaneously accessing one or more stack elements by multiple functional units using real stack addresses

DATE-ISSUED: November 14, 2000

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Petrick; Bruce	Sunnyvale	CA		

US-CL-CURRENT: 712/202; 711/109, 711/200, 712/200, 712/215, 712/216, 712/217, 712/228, 712/23

ABSTRACT:

Embodiments of the present invention provide a stack renaming method and apparatus for stack based processors. Using principles of the present invention, a stack can be accessed simultaneously by one or more functional units in a stack processor. The stack apparatus includes a stack renaming unit capable of renaming a logical stack address to a real stack address. Each logical stack address corresponds to a

storage element in the stack renaming unit which stores a real stack address. A circular counter is used in the stack renaming unit to sequentially cycle through each of the logical stack addresses. The real stack addresses corresponding to each of the logical stack addresses can be stored out of order in the stack renaming unit. A stack control unit is coupled to the stack renaming unit and provides one or more control signals to the stack renaming unit and coordinates the operation of the stack renaming unit within the stack apparatus. A stack memory circuit coupled to the stack renaming unit receives the real stack address and generates a data value to be used in a functional unit. The real address is provided to a functional unit by a real address output port coupled to the stack renaming unit. As long as there are no data dependencies or structural dependences, a functional unit is able to operate on the data value associated with the real address while other functional units may be accessing and using other stack values.

11 Claims, 13 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 11

☐ 48. Document ID: US 6148346 A

L4: Entry 48 of 58 File: USPT Nov 14, 2000

US-PAT-NO: 6148346
DOCUMENT-IDENTIFIER: US 6148346 A

TITLE: Dynamic device driver

DATE-ISSUED: November 14, 2000

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Hanson; Gordon L.	Kent	WA		

US-CL-CURRENT: 719/321; 345/961

ABSTRACT:

A data communication system for allowing communication between various devices and various operating systems across various types of networking systems. The data communication system includes an host computer system 10 with a display device 15 and a processor 17 for generating signals for the display device, at least one peripheral device 56, and a dynamic device driver 42 for allowing two-way communication between the peripheral device and the host computer system. The dynamic device driver includes an operating system specific portion 33, configured for the operating system of the host computer system, and an operating system independent device driver portion 34, configured for the peripheral device. The operating system includes a linking mechanism 19 for allowing two-way communication between the operating system specific portion and the operating system independent device driver portion, thus allowing two-way communication between the processor and the peripheral device.

32 Claims, 22 Drawing figures

Exemplary Claim Number: 1
Number of Drawing Sheets: 12

Full | Title | Citation | Front | Review | Classification | Date | Reference | Claims | KWIC | Draw. D

☐ 49. Document ID: US 6122638 A

L4: Entry 49 of 58 File: USPT Sep 19, 2000

US-PAT-NO: 6122638
DOCUMENT-IDENTIFIER: US 6122638 A

TITLE: Object-oriented processor and method for caching intermediate data in an object-oriented processor

DATE-ISSUED: September 19, 2000

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Huber; Gary Douglas	Austin	TX		
McCauley; Donald William	Austin	TX		

US-CL-CURRENT: 707/103Y; 711/103, 719/313

ABSTRACT:

An object-oriented processor and method of operating such a processor are disclosed. According to the method, in response to receiving a first instruction that references a first object having both data and at least a first method associated therewith, an address of the first method is calculated. In addition, at least one pointer is cached that indicates (possibly together with other pointers) the address of the first method. In response to receipt of a subsequent second instruction that references a second object having both data and one or more methods associated therewith, where the one or more methods include the first method, the address of a second method associated with the second object is determined by reference to the cached pointer. In a preferred embodiment of the present invention, the first and second methods comprise the same method, and the cached pointer indicates the entry point of that single method.

21 Claims, 5 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 5

Full | Title | Citation | Front | Review | Classification | Date | Reference | Claims | KWIC | Draw. D

☐ 50. Document ID: US 6118940 A

L4: Entry 50 of 58 File: USPT Sep 12, 2000

US-PAT-NO: 6118940
DOCUMENT-IDENTIFIER: US 6118940 A

TITLE: Method and apparatus for benchmarking byte code sequences

DATE-ISSUED: September 12, 2000

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Alexander, III; William Preston	Austin	TX		
Berry; Robert Francis	Austin	TX		
Hussain; Riaz	Austin	TX		
Kilpatrick; Paul Jerome	Austin	TX		
Urquhart; Robert John	Austin	TX		

US-CL-CURRENT: 717/127; 714/38, 717/147, 717/148, 718/1

ABSTRACT:

Method and apparatus for creating benchmark programs for the analysis of java virtual machine implementations are implemented. Java applications and applets are compiled into an intermediate code referred to as byte code. The Java byte code forms the machine code for the Java Virtual Machine. The Java Virtual Machine running on top of a hardware platform translates the byte code into native machine code for execution on the hardware platform on which the Java Virtual Machine is running. The performance of a Java Virtual Machine is improved by the use of a so-called "just in time" (JIT) compiler that translates commonly occurring sequences of bytes codes in the native instruction sequences which are then stored for later execution. Critical to the performance of the JIT is the ability of the JIT to optimally compile for the most commonly occurring sequences of byte codes. The method and apparatus for creating benchmark programs provides a means for performance measurements with respect to such sequences.

43 Claims, 9 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 6

Terms	Documents
L3 and (dependency OR dependencies)	58

Hit List

Clear | Generate Collection | Print | Fwd Refs | Bkwd Refs | Generate OACS

Search Results - Record(s) 51 through 58 of 58 returned.

☑ 51. Document ID: US 6092120 A

L4: Entry 51 of 58 File: USPT Jul 18, 2000

US-PAT-NO: 6092120
DOCUMENT-IDENTIFIER: US 6092120 A

TITLE: Method and apparatus for timely delivery of a byte code and serialized objects stream

DATE-ISSUED: July 18, 2000

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Swaminathan; Viswanathan	Union City	CA		
Fernando; Gerard	Mountain View	CA		
Speer; Michael	Mtn View	CA		

US-CL-CURRENT: 709/247; 709/231, 709/236

ABSTRACT:

A method and apparatus for timely delivery of classes and objects is provided. A header comprising timing information is attached to said classes and/or objects. A "start loading" time and a "load by" time are specified in the header. Other classes and/or objects to be loaded are also specified in the header. Optional compression, security, and/or error resilience schemes are also specified in the header. A process for creating the header and attaching it to a class or object is provided. A process for receiving and processing a class or object with an attached header is provided. Embodiments of the invention allow timely delivery of classes and/or objects over a wide variety of transport mechanisms, including unreliable transport mechanisms and those lacking any guarantees of timely delivery.

23 Claims, 9 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 9

Full | Title | Citation | Front | Review | Classification | Date | Reference | Claims | KWIC | Draw De

☑ 52. Document ID: US 6081665 A

L4: Entry 52 of 58 File: USPT Jun 27, 2000

US-PAT-NO: 6081665
DOCUMENT-IDENTIFIER: US 6081665 A

TITLE: Method for efficient soft real-time execution of portable byte code computer programs

DATE-ISSUED: June 27, 2000

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Nilsen; Kelvin D.	Ames	IA		
Mitra; Simanta	Ames	IA		
Lee; Steven J.	Slater	IA		

US-CL-CURRENT: 717/116; 717/118, 717/148

ABSTRACT:

The invention is a method for use in executing portable virtual machine computer programs under real-time constraints. The invention includes a method for implementing a single abstract virtual machine execution stack with multiple independent stacks in order to improve the efficiency of distinguishing memory pointers from non-pointers. Further, the invention includes a method for rewriting certain of the virtual machine instructions into a new instruction set that more efficiently manipulates the multiple stacks. Additionally, using the multiple-stack technique to identify pointers on the run-time stack, the invention includes a method for performing efficient defragmenting real-time garbage collection using a mostly stationary technique. The invention also includes a method for efficiently mixing a combination of byte-code, native, and JIT-translated methods in the implementation of a particular task, where byte-code methods are represented in the instruction set of the virtual machine, native methods are written in a language like C and represented by native machine code, and JIT-translated methods result from automatic translation of byte-code methods into the native machine code of the host machine. Also included in the invention is a method to implement a real-time task dispatcher that supports arbitrary numbers of real-time task priorities given an underlying real-time operating system that supports at least three task priority levels. Finally, the invention includes a method to analyze and preconfigure virtual memory programs so that they can be stored in ROM memory prior to program.

89 Claims, 100 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 46

☑ 53. Document ID: US 6078744 A

L4: Entry 53 of 58 File: USPT Jun 20, 2000

US-PAT-NO: 6078744
DOCUMENT-IDENTIFIER: US 6078744 A

TITLE: Method and apparatus for improving compiler performance during subsequent compilations of a source program

DATE-ISSUED: June 20, 2000

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Wolczko; Mario I.	Palo Alto	CA		
Ungar; David M.	Palo Alto	CA		

US-CL-CURRENT: 717/153; 700/28, 717/116, 717/145, 717/146, 717/159

ABSTRACT:

Apparatus, methods, and computer program products are disclosed for improving the performance of subsequent compilations of a source program. The initial compilation of the source program journals computationally expensive intermediate compilation data. Subsequent compilations, instead of recomputing this data, uses the intermediate compilation data from the journal. Thus the compiler has improved performance during subsequent compilations of the source program. One aspect of the invention applies to dynamic compilers to reduce the startup delay caused by compiling the source program when the program is invoked.

8 Claims, 8 Drawing figures
Exemplary Claim Number: 8
Number of Drawing Sheets: 8

Full | Title | Citation | Front | Review | Classification | Date | Reference | Claims | KWMC | Draw. De

☑ 54. Document ID: US 6026485 A

L4: Entry 54 of 58 File: USPT Feb 15, 2000

US-PAT-NO: 6026485
DOCUMENT-IDENTIFIER: US 6026485 A

TITLE: Instruction folding for a stack-based machine

DATE-ISSUED: February 15, 2000

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
O'Connor; James Michael	Mountain View	CA		
Tremblay; Marc	Palo Alto	CA		

US-CL-CURRENT: 712/226; 712/202, 712/209, 712/217, 712/26, 717/148

ABSTRACT:

An instruction decoder allows the folding away of JAVA virtual machine instructions pushing an operand onto the top of a stack merely as a precursor to a second JAVA virtual machine instruction which operates on the top of stack operand. Such an instruction decoder identifies foldable instruction sequences and supplies an execution unit with a single equivalent folded operation thereby reducing processing cycles otherwise required for execution of multiple operations

corresponding to the multiple instructions of the folded instruction sequence. Instruction decoder embodiments described herein provide for folding of two, three, four, or more instruction folding. For example, in one instruction decoder embodiment described herein, two load instructions and a store instruction can be folded into execution of operation corresponding to an instruction appearing therebetween in the instruction sequence.

26 Claims, 25 Drawing figures
Exemplary Claim Number: 24
Number of Drawing Sheets: 15

Full | Title | Citation | Front | Review | Classification | Date | Reference | Claims | KWMC | Draw. De

☐ 55. Document ID: US 5966542 A

L4: Entry 55 of 58 File: USPT Oct 12, 1999

US-PAT-NO: 5966542
DOCUMENT-IDENTIFIER: US 5966542 A

TITLE: Method and system for loading classes in read-only memory

DATE-ISSUED: October 12, 1999

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Tock; Theron D.	Sunnyvale	CA		

US-CL-CURRENT: 717/166; 713/1, 713/2, 717/116, 717/163

ABSTRACT:

A method and system for providing an executable module having an address space for storing program data that is to reside in a read-only storage medium and an address space for storing program data that is to reside in a random access memory is herein described. The executable module represents Java classes that are structured for dynamic class loading. A static class loader is used to modify the class structure to accommodate static loading. The static class loader also identifies methods that contain unresolved symbolic references and data that varies during the execution of the module. These methods and data are identified in order to place them in the address space that resides in the random access memory. The static loader is beneficial in a distributed computing environment having a client computer that has little or no secondary storage thereby requiring applications to run entirely in random access memory. By utilizing a read-only memory to store statically loadable classes, the random access memory is left available for other uses.

6 Claims, 13 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 12

Full | Title | Citation | Front | Review | Classification | Date | Reference | Claims | KWMC | Draw. De

☐ 56. Document ID: US 5892966 A

L4: Entry 56 of 58 File: USPT Apr 6, 1999

US-PAT-NO: 5892966
DOCUMENT-IDENTIFIER: US 5892966 A

TITLE: Processor complex for executing multimedia functions

DATE-ISSUED: April 6, 1999

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Petrick; Bruce E.	Sunnyvale	CA		
Patel; Mukesh	Fremont	CA		

US-CL-CURRENT: 712/36; 707/104.1, 712/22, 715/500.1

ABSTRACT:

A computer processor complex including a hardware processor coupled to a multimedia coprocessor is provided. This computer processor complex is capable of separately processing a stream of non-multimedia instructions in addition to a stream of multimedia instructions such as are used in MPEG audio and video. The computer processor complex includes a visible register set including registers for a program counter and a data pointer. The program counter is used to hold the address in memory where the multimedia instructions are located and the data pointer indicates where the data, corresponding to these multimedia instructions, is located in memory. A hardware processor is coupled to a first bidirectional port on the visible register set and a multimedia coprocessor is coupled to a second bidirectional port on the visible register set. The bidirectional ports allow the hardware processor and the coprocessor to exchange data and status information typically using an interrupt based communication mechanism. A main memory device is also coupled to the hardware processor over bidirectional port and coupled to the multimedia processor over a second bidirectional port. This arrangement allows the hardware processor and the coprocessor to share main memory and load separate instruction streams from main memory.

15 Claims, 8 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 8

Full | Title | Citation | Front | Review | Classification | Date | Reference | Claims | KWIC | Draw De

☐ 57. Document ID: US 5815718 A

L4: Entry 57 of 58 File: USPT Sep 29, 1998

US-PAT-NO: 5815718
DOCUMENT-IDENTIFIER: US 5815718 A

TITLE: Method and system for loading classes in read-only memory

DATE-ISSUED: September 29, 1998

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Tock; Theron D.	Sunnyvale	CA		

US-CL-CURRENT: 717/166; 717/118, 719/331

ABSTRACT:

A method and system for providing an executable module having an address space for storing program data that is to reside in a read-only storage medium and an address space for storing program data that is to reside in a random access memory is herein described. The executable module represents Java classes that are structured for dynamic class loading. A static class loader is used to modify the class structure to accommodate static loading. The static class loader also identifies methods that contain unresolved symbolic references and data that varies during the execution of the module. These methods and data are identified in order to place them in the address space that resides in the random access memory. The static loader is beneficial in a distributed computing environment having a client computer that has little or no secondary storage thereby requiring applications to run entirely in random access memory. By utilizing a read-only memory to store statically loadable classes, the random access memory is left available for other uses.

20 Claims, 14 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 12

☐ 58. Document ID: US 5787431 A

L4: Entry 58 of 58 File: USPT Jul 28, 1998

US-PAT-NO: 5787431
DOCUMENT-IDENTIFIER: US 5787431 A

TITLE: Database development system with methods for java-string reference lookups of column names

DATE-ISSUED: July 28, 1998

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Shaughnessy; Steven T.	Scotts Valley	CA		

US-CL-CURRENT: 707/100; 370/392, 370/477, 707/1, 707/3, 711/206, 713/500, 717/102

ABSTRACT:

A Java-based rapid application development (RAD) environment for creating applications providing named-based programmatic access to information from columns in databases is described. For increasing the efficiency by which named-based references to database columns are processed by application programs, the system provides methodology for rapid lookups of column names, using a reference cache storing 32-bit references to immutable strings (e.g., Java strings). The reference cache is preferably constructed as a least-recently allocated cache, thereby allowing allocation to occur in a round-robin fashion, with the oldest item allocated being the first item bumped from cache when the cache overflows. Each cache entry stores a reference (e.g., four-byte pointer or handle to a string) and an ordinal entry (i.e. the corresponding database ordinal). As a reference to a particular database column occurs during execution of a program, the reference cache fills with a reference to that column name as well as the corresponding column ordinal. Accordingly, program execution proceeds with comparison of existing items in the cache, using a sequence of rapid, in-line comparisons involving simple data types (e.g., 32-bit references for the column name string). This approach minimizes the need to perform hash lookups or string comparison operations.

20 Claims, 6 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 4

Terms	Documents
L3 and (dependency OR dependencies)	58

Refine Search

Search Results -

Terms	Documents
L3 and (dependency OR dependencies)	58

Database:
US Pre-Grant Publication Full-Text Database
US Patents Full-Text Database
US OCR Full-Text Database
EPO Abstracts Database
JPO Abstracts Database
Derwent World Patents Index
IBM Technical Disclosure Bulletins

Search: L4

Refine Search

Recall Text | Clear | Interrupt

Search History

DATE: Saturday, May 29, 2004 Printable Copy Create Case

Set Name side by side	Query	Hit Count	Set Name result set
DB=USPT; PLUR=NO; OP=OR			
L4	L3 and (dependency OR dependencies)	58	L4
L3	L1 and JAVA	337	L3
L2	L1 and compiler	353	L2
L1	JIT OR (just ADJ in adj time) OR (just-in-time) AND (compiler OR compiling)	672	L1

END OF SEARCH HISTORY

PORTAL
US Patent & Trademark Office
Subscribe (Full Service) Register (Limited Service, Free) Login
Search: The ACM Digital Library The Guide
JIT dependency reference
SEARCH
THE ACM DIGITAL LIBRARY
Feedback Report a problem Satisfaction survey

Terms used **JIT dependency reference** Found **13,465** of **134,837**

Sort results by: relevance
Display results: expanded form
Save results to a Binder
Search Tips
Open results in a new window

Try an Advanced Search
Try this search in The ACM Guide

Results 81 - 100 of 200 Result page: previous 1 2 3 4 **5** 6 7 8 9 10 next
Best 200 shown Relevance scale

81 Logic clause analysis for delay optimization
Berhard Rohfleisch, Bernd Wurth, Kurt Antreich
January 1995 **Proceedings of the 32nd ACM/IEEE conference on Design automation conference**
Full text available: pdf(210.28 KB) Additional Information: full citation, references, citings, index terms

82 Technical papers: testing II: Data flow testing as model checking
Hyoung Seok Hong, Sung Deok Cha, Insup Lee, Oleg Sokolsky, Hasan Ural
May 2003 **Proceedings of the 25th international conference on Software engineering**
Full text available: pdf(1.00 MB) Additional Information: full citation, abstract, references
Publisher Site

This paper presents a model checking-based approach to data flow testing. We characterize data flow oriented coverage criteria in temporal logic such that the problem of test generation is reduced to the problem of finding witnesses for a set of temporal logic formulas. The capability of model checkers to construct witnesses and counterexamples allows test generation to be fully automatic. We discuss complexity issues in minimal cost test generation and describe heurstic test generation algorith ...

83 Using indexed data structures for program specialization
Jung Gyu Park, Myong-Soon Park
September 2002 **Proceedings of the ASIAN symposium on Partial evaluation and semantics-based program manipulation**
Full text available: pdf(201.23 KB) Additional Information: full citation, abstract, references, index terms

Given a program and values of static (fixed) inputs, program specialization generates an optimized version of the program that only requires dynamic (run-time) inputs. It has been an useful tool for such areas as operating systems, multimedia applications, and scientific applications. However, the size of specialized code may grow up exponentially which makes program specialization impractical for many applications. In this paper, we present a mechanism to address this problem by using indexed d ...

Keywords: automated software engineering, partial evaluation, program specialization, staged computation optimization

84 TED models for ATM internetworks

Kalyan Perumalla, Matthew Andrews, Sandeep Bhatt

March 1998 **ACM SIGMETRICS Performance Evaluation Review**, Volume 25 Issue 4

Full text available: pdf(768.18 KB) Additional Information: full citation, abstract, index terms

We describe our experiences designing and implementing a virtual PNNI network testbed. The network elements and signaling protocols modeled are consistent with the ATM Forum *PNNI* draft specifications. The models will serve as a high-fidelity testbed of the transport and network layers for simulation-based studies of the scalability and performance of PNNI protocols.Our models are written in the new network description language TeD which offers two advantages. First, the testbed d ...

85 Performance analysis and optimization of schedules for conditional and loop-intensive specifications

Subhrajit Bhattacharya, Sujit Dey, Franc Brglez

June 1994 **Proceedings of the 31st annual conference on Design automation conference**

Full text available: pdf(121.98 KB) Additional Information: full citation, references, citings, index terms

86 CommonLoops: merging Lisp and object-oriented programming

Daniel G. Bobrow, Kenneth Kahn, Gregor Kiczales, Larry Masinter, Mark Stefik, Frank Zdybel

June 1986 **ACM SIGPLAN Notices , Conference proceedings on Object-oriented programming systems, languages and applications**, Volume 21 Issue 11

Full text available: pdf(1.06 MB) Additional Information: full citation, abstract, references, citings, index terms

CommonLoops blends object-oriented programming smoothly and tightly with the procedure-oriented design of Lisp. Functions and methods are combined in a more general abstraction. Message passing is invoked via normal Lisp function call. Methods are viewed as partial descriptions of procedures. Lisp data types are integrated with object classes. With these integrations, it is easy to incrementally move a program between the procedure and object-oriented styles.

87 Performance measurement of dynamically compiled Java executions

Tia Newhall, Barton P. Miller

June 1999 **Proceedings of the ACM 1999 conference on Java Grande**

Full text available: pdf(1.54 MB) Additional Information: full citation, references, index terms

Keywords: Java, dynamic compilation, performance profiling tool

88 Java consistency: nonoperational characterizations for Java memory behavior

Alex Gontmakher, Assaf Schuster

November 2000 **ACM Transactions on Computer Systems (TOCS)**, Volume 18 Issue 4

Full text available: pdf(305.72 KB) Additional Information: full citation, abstract, references, citings, index terms

The Java Language Specification (JLS) [Gosling et al. 1996] provides an operational definition for the consistency of shared variables. The definition remains unchanged in the JLS 2nd edition, currently under peer review, which relies on a specific abstract machine as its underlying model, is very complicated. Several subsequent works have tried to simplify and formalize it. However, these revised definitions are also operational, and thus have failed to highlight the intuition behind the o ...

Keywords: Java memory models, multithreading, nonoperational specification

89 An integrated metric for video QoS
Nalini Venkatasubramanian, Klara Nahrstedt
November 1997 **Proceedings of the fifth ACM international conference on Multimedia**
Full text available: pdf(1.99 MB) Additional Information: full citation, references, citings, index terms

90 A component repository for learning objects: a progress report
Jean R. Laleuf, Anne Morgan Spalter
January 2001 **Proceedings of the first ACM/IEEE-CS joint conference on Digital libraries**
Full text available: pdf(699.99 KB) Additional Information: full citation, abstract, references, index terms

We believe that an important category of SMET digital library content will be highly interactive, explorable microworlds for teaching science, mathematics, and engineering concepts. Such environments have proved extraordinarily time-consuming and difficult to produce, however, threatening the goals of widespread creation and use. One proposed solution for accelerating production has been the creation of repositories of reusable software components or learning objects. Programmer ...

Keywords: NSDL, components, design, digital library, education, learning objects, reuse, software engineering, standards

91 Using complete-1-distinguishability for FSM equivalence checking
Pranav Ashar, Aarti Gupta, Sharad Malik
October 2001 **ACM Transactions on Design Automation of Electronic Systems (TODAES)**, Volume 6 Issue 4
Full text available: pdf(286.12 KB) Additional Information: full citation, abstract, references, index terms

This article introduces the notion of a Complete-1-Distinguishability (C-1-D) property for simplifying equivalence checking of finite state machines (FSMs). When a specification machine has the C-1-D property, the traversal of the product machine can be eliminated. Instead, a much simpler check suffices. The check consists of first obtaining a 1-equivalence mapping between the individually reachable states of the specification and the implementation machines, and then checking that it is a bisim ...

Keywords: Bisimulation relation, complete-1-distinguishability, equivalence checking, finite state machine equivalence, sequential logic synthesis

92 Design of heterogeneous ICs for mobile and personal communication systems
Gert Goossens, Ivo Bolsens, Bill Lin, Francky Catthoor
November 1994 **Proceedings of the 1994 IEEE/ACM international conference on Computer-aided design**
Full text available: pdf(1.04 MB) Additional Information: full citation, abstract, references, citings, index terms

Mobile and personal communication systems form key market areas for the electronics industry of the nineties. Stringent requirements in terms of flexibility, performance and power dissipation, are driving the development of integrated circuits into the direction of heterogeneous single-chip solutions. New IC architectures are emerging which contain the core of a powerful programmable processor, complemented with dedicated hardware,

memory and interface structures. In this tutorial we will d ...

93 Automatic verification of pipelined microprocessors
Vishal Bhagwati, Srinivas Devadas
June 1994 **Proceedings of the 31st annual conference on Design automation conference**
Full text available: pdf(61.53 KB) Additional Information: full citation, references, citings, index terms

94 Fast simulation methods for highly dependable systems
Marvin K. Nakayama
December 1994 **Proceedings of the 26th conference on Winter simulation**
Full text available: pdf(865.52 KB) Additional Information: full citation, references, citings, index terms

95 Josh: an open AspectJ-like language
Shigeru Chiba, Kiyoshi Nakagawa
March 2004 **Proceedings of the 3rd international conference on Aspect-oriented software development**
Full text available: pdf(1.09 MB) Additional Information: full citation, abstract, references

Although aspect-oriented programming (AOP) is becoming widely used, the design of the pointcut language and the generic and reusable description of advice are still research topics. To address these topics, this paper presents *Josh*, which is our new AspectJ-like language with an extensible pointcut language and a few mechanisms for generic description. The extensible pointcut language is based on the idea of open compiler. Since Josh allows defining a new pointcut designator in Java, the ...

Keywords: extensibility, generic description, pointcut

96 Modular type-based reverse engineering of parameterized types in Java code
Dominic Duggan
October 1999 **ACM SIGPLAN Notices , Proceedings of the 14th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications**, Volume 34 Issue 10
Full text available: pdf(1.77 MB) Additional Information: full citation, abstract, references, citings, index terms

An analysis is provided for Java programs that reverse engineers parameterized types into existing Java code. This analysis propagates precise type information about the contents of container objects. As an application, the analysis can be used to justify the safe removal of downcasts that are guaranteed to succeed. Another application is in automatically reverse engineering parameterized types into existing Java libraries, so that they can be used in Java dialects with parameterized types. ...

97 Optimal histograms for limiting worst-case error propagation in the size of join results
Yannis E. Ioannidis, Stavros Christodoulakis
December 1993 **ACM Transactions on Database Systems (TODS)**, Volume 18 Issue 4
Full text available: pdf(2.44 MB) Additional Information: full citation, references, citings, index terms

Keywords: histograms, join size estimation, query optimization, vector majorization

98 Applying component technology to improve global supply chain network management
Gek Woo Tan, Michael J. Shaw
December 1998 **Proceedings of the international conference on Information systems**
Full text available: pdf(135.15 KB) Additional Information: full citation, references, citings, index terms

Keywords: component technology, electronic commerce, simulation and modeling IS, supply chain management

99 JavaGenes and Condor: cycle-scavenging genetic algorithms
Al Globus, Eric Langhirt, Miron Livny, Ravishankar Ramamurthy, Marvin Solomon, Steve Traugott
June 2000 **Proceedings of the ACM 2000 conference on Java Grande**
Full text available: pdf(603.01 KB) Additional Information: full citation, references, index terms

Keywords: Condor, Java, genetic algorithms

100 Design and specification of embedded systems in Java using successive, formal refinement
James Shin Young, Josh MacDonald, Michael Shilman, Abdallah Tabbara, Paul Hilfinger, A. Richard Newton
May 1998 **Proceedings of the 35th annual conference on Design automation conference**
Full text available: pdf(256.51 KB) Additional Information: full citation, references, citings, index terms

Results 81 - 100 of 200 Result page: previous 1 2 3 4 **5** 6 7 8 9 10 next

The ACM Portal is published by the Association for Computing Machinery. Copyright © 2004 ACM, Inc.
Terms of Usage Privacy Policy Code of Ethics Contact Us

Useful downloads: Adobe Acrobat QuickTime Windows Media Player Real Player

ACM PORTAL
US Patent & Trademark Office
Subscribe (Full Service) Register (Limited Service, Free) Login
Search: The ACM Digital Library The Guide
JIT dependency reference
SEARCH
THE ACM DIGITAL LIBRARY
Feedback Report a problem Satisfaction survey

Terms used **JIT dependency reference** Found **13,465** of **134,837**

Sort results by: relevance
Display results: expanded form
Save results to a Binder
Search Tips
Open results in a new window
Try an Advanced Search
Try this search in The ACM Guide

Results 61 - 80 of 200 Result page: previous 1 2 3 **4** 5 6 7 8 9 10 next
Best 200 shown Relevance scale

61 Partial behavioral reflection: spatial and temporal selection of reification
Éric Tanter, Jacques Noyé, Denis Caromel, Pierre Cointe
October 2003 **ACM SIGPLAN Notices , Proceedings of the 18th ACM SIGPLAN conference on Object-oriented programing, systems, languages, and applications**, Volume 38 Issue 11
Full text available: pdf(261.44 KB) Additional Information: full citation, abstract, references, index terms

Behavioral reflection is a powerful approach for adapting the behavior of running applications. In this paper we present and motivate *partial behavioral reflection*, an approach to more efficient and flexible behavioral reflection. We expose the *spatial* and *temporal* dimensions of such reflection, and propose a model of partial behavioral reflection based on the notion of *hooksets*. In the context of Java, we describe a reflective architecture offering appropriate interf ...

Keywords: aspect-oriented programming, open systems, reflection

62 Hybrid dynamic data race detection
Robert O'Callahan, Jong-Deok Choi
June 2003 **ACM SIGPLAN Notices , Proceedings of the ninth ACM SIGPLAN symposium on Principles and practice of parallel programming**, Volume 38 Issue 10
Full text available: pdf(158.47 KB) Additional Information: full citation, abstract, references, index terms

We present a new method for dynamically detecting potential data races in multithreaded programs. Our method improves on the state of the art in accuracy, in usability, and in overhead. We improve accuracy by combining two previously known race detection techniques -- *lockset-based detection and happens-before-based detection* -- to obtain fewer false positives than lockset-based detection alone. We enhance usability by reporting more information about detected races than any previous dyna ...

Keywords: Java, dynamic race detection, happens-before, lockset hybrid

63 Manageable object-oriented development: abstraction, decomposition, and modeling
John A. Anderson, John D. Sheffler, Elaine S. Ward
December 1991 **Proceedings of the conference on TRI-Ada '91: today's accomplishments; tomorrow's expectations**
Full text available: pdf(1.70 MB) Additional Information: full citation, references, citings, index terms

64 TGFF: task graphs for free
Robert P. Dick, David L. Rhodes, Wayne Wolf
March 1998 **Proceedings of the 6th international workshop on Hardware/software codesign**
Full text available: pdf(102.55 KB) Publisher Site
Additional Information: full citation, references, citings, index terms

65 Java bytecode to native code translation: the caffeine prototype and preliminary results
Cheng-Hsueh A. Hsieh, John C. Gyllenhaal, Wen-mei W. Hwu
December 1996 **Proceedings of the 29th annual ACM/IEEE international symposium on Microarchitecture**
Full text available: pdf(1.03 MB) Publisher Site
Additional Information: full citation, abstract, references, citings, index terms

The Java bytecode language is emerging as a software distribution standard. With major vendors committed to porting the Java run-time environment to their platforms, programs in Java bytecode are expected to run without modification on multiple platforms. These first generation run-time environments rely on an interpreter to bridge the gap between the bytecode instructions and the native hardware. This interpreter approach is sufficient for specialized applications such as Internet browsers wher ...

66 A high performance Erlang system
Erik Johansson, Mikael Pettersson, Konstantinos Sagonas
September 2000 **Proceedings of the 2nd ACM SIGPLAN international conference on Principles and practice of declarative programming**
Full text available: pdf(320.62 KB) Additional Information: full citation, references, citings, index terms

67 JMTP: an architecture for exploiting concurrency in embedded Java applications with real-time considerations
Rachid Helaihel, Kunle Olukotun
November 1999 **Proceedings of the 1999 IEEE/ACM international conference on Computer-aided design**
Full text available: pdf(139.94 KB) Additional Information: full citation, abstract, references, index terms

Using Java in embedded systems is plagued by problems of limited runtime performance and unpredictable runtime behavior. The Java Multi-Threaded Processor (JMTP) provides solutions to these problems. The JMTP architecture is a single chip containing an off-the-shelf general purpose processor core coupled with an array of Java Thread Processors (JTPs). Performance can be improved using this architecture by exploiting coarse-grained parallelism in the application. These performance im ...

68 Information technology and physical space
Henry C. Lucas
November 2001 **Communications of the ACM**, Volume 44 Issue 11
Full text available: pdf(122.14 KB) html(42.79 KB)
Additional Information: full citation, references, index terms

69 DDD papers: XAspects: an extensible system for domain-specific aspect languages
Macneil Shonle, Karl Lieberherr, Ankit Shah
October 2003 **Companion of the 18th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications**
Full text available: pdf(218.84 KB) Additional Information: full citation, abstract, references, index terms

Current general aspect-oriented programming solutions fall short of helping the problem of separation of concerns for several concern domains. Because of this limitation good solutions for these concern domains do not get used and the opportunity to benefit from separation of these concerns is missed. By using XAspects, a plug-in mechanism for domain-specific aspect languages, separation of concerns can be achieved at a level beyond what is possible for object-oriented programming languages. As ...

Keywords: aspect-oriented programming, domain-specific languages, generative programming, language extensions

70 Virtual machine support for dynamic join points
Christoph Bockisch, Michael Haupt, Mira Mezini, Klaus Ostermann
March 2004 **Proceedings of the 3rd international conference on Aspect-oriented software development**
Full text available: pdf(1.19 MB) Additional Information: full citation, abstract, references

A widespread implementation approach for the join point mechanism of aspect-oriented languages is to instrument areas in code that match the static part of pointcut designators, inserting dynamic checks for that part of matching that depends on run-time conditions, if needed. For performance reasons, such dynamic checks should be avoided whenever possible. One way to do so is to postpone weaving of advice calls until run-time, when conditions determining the emergence of join points hold. This c ...

71 Bytecode fetch optimization for a Java interpreter
Kazunori Ogata, Hideaki Komatsu, Toshio Nakatani
October 2002 **Tenth international conference on architectural support for programming languages and operating systems on Proceedings of the 10th international conference on architectural support for programming languages and operating systems (ASPLOS-X)**, Volume 36 , 37 , 30 Issue 5 , 10 , 5
Full text available: pdf(1.16 MB) Additional Information: full citation, abstract, references, index terms

Interpreters play an important role in many languages, and their performance is critical particularly for the popular language Java. The performance of the interpreter is important even for high-performance virtual machines that employ just-in-time compiler technology, because there are advantages in delaying the start of compilation and in reducing the number of the target methods to be compiled. Many techniques have been proposed to improve the performance of various interpreters, but none of ...

Keywords: Java, PowerPC, bytecode interpreter, performance, pipelined interpreter, stack caching, superscalar processor

72 MultiMedia: Context perception in video-based hypermedia spaces
Teresa Chambel, Nuno Guimarães
June 2002 **Proceedings of the thirteenth ACM conference on Hypertext and hypermedia**
Full text available: pdf(648.49 KB) Additional Information: full citation, abstract, references, citings, index terms

Multimedia hypertext has grown from the basic addition of dynamic media only at "leaf"

nodes of the hypertext, to higher structured attempts to compose and integrate the different media. One of the core problems in this evolution has been, and still is, the construction and perception of context, making explicit which part of a presentation is relevant when media elements are integrated. The search for contextualized integration of video material with other sources of information has emerged fro ...

Keywords: HTIMEL, cognitive processes, design, education, entertainment, hypervideo, integration in context, interactive video and TV, link awareness

73 jRapture: A Capture/Replay tool for observation-based testing

John Steven, Pravir Chandra, Bob Fleck, Andy Podgurski

August 2000 **ACM SIGSOFT Software Engineering Notes , Proceedings of the International Symposium on Software Testing and Analysis**, Volume 25 Issue 5

Full text available: pdf(403.58 KB) Additional Information: full citation, abstract, references, citings, index terms

We describe the design of jRapture: a tool for capturing and replaying Java program executions in the field. jRapture works with Java binaries (byte code) and any compliant implementation of the Java virtual machine. It employs a lightweight, transparent capture process that permits unobtrusive capture of a Java programs executions. jRapture captures interactions between a Java program and the system, including GUI, file, and console inputs, among other types, and on replay it presents eac ...

Keywords: Java, capture/replay, execution profiling, observation-based testing, software testing

74 Performance-driven scheduling with bit-level chaining

Sanghun Park, Kiyoung Choi

June 1999 **Proceedings of the 36th ACM/IEEE conference on Design automation conference**

Full text available: pdf(132.33 KB) Additional Information: full citation, references, citings, index terms

75 Binary translation and architecture convergence issues for IBM system/390

Michael Gschwind, Kemal Ebcioğlu, Erik Altman, Sumedh Sathaye

May 2000 **Proceedings of the 14th international conference on Supercomputing**

Full text available: pdf(1.44 MB) Additional Information: full citation, abstract, references, index terms

We describe the design issues in an implementation of the ESA/390 architecture based on binary translation to a very long instruction word (VLIW) processor. During binary translation, complex ESA/390 instructions are decomposed into instruction "primitives" which are then scheduled onto a wide-issue machine. The aim is to achieve high instruction level parallelism due to the increased scheduling and optimization opportunities which can be exploited by binary translation software ...

76 Thin locks: featherweight synchronization for Java

David F. Bacon, Ravi Konuru, Chet Murthy, Mauricio Serrano

May 1998 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 1998 conference on Programming language design and implementation**, Volume 33 Issue 5

Full text available: pdf(1.30 MB) Additional Information: full citation, abstract, references, citings, index terms

Language-supported synchronization is a source of serious performance problems in many Java programs. Even single-threaded applications may spend up to half their time

performing useless synchronization due to the thread-safe nature of the Java libraries. We solve this performance problem with a new algorithm that allows lock and unlock operations to be performed with only a few machine instructions in the most common cases. Our locks only require a partial word per object, and were implemented ...

77 Energy aware design: Optimizing pipelines for power and performance

Viji Srinivasan, David Brooks, Michael Gschwind, Pradip Bose, Victor Zyuban, Philip N. Strenski, Philip G. Emma

November 2002 **Proceedings of the 35th annual ACM/IEEE international symposium on Microarchitecture**

Full text available: pdf(1.24 MB) Publisher Site Additional Information: full citation, abstract, references, index terms

During the concept phase and definition of next generation high-end processors, power and performance will need to be weighted appropriately to deliver competitive cost/performance. It is not enough to adopt a CPI-centric view alone in early-stage definition studies. One of the fundamental issues confronting the architect at this stage is the choice of pipeline depth and target frequency. In this paper we present an optimization methodology that starts with an analytical power-performance model ...

78 Partial method compilation using dynamic profile information

John Whaley

October 2001 **ACM SIGPLAN Notices , Proceedings of the 16th ACM SIGPLAN conference on Object oriented programming, systems, languages, and applications**, Volume 36 Issue 11

Full text available: pdf(1.73 MB) Additional Information: full citation, abstract, references, citings, index terms

The traditional tradeoff when performing dynamic compilation is that of fast compilation time versus fast code performance. Most dynamic compilation systems for Java perform selective compilation and/or optimization at a method granularity. This is the not the optimal granularity level. However, compiling at a sub-method granularity is thought to be too complicated to be practical. This paper describes a straightforward technique for performing compilation and optimizations at a finer, sub-metho ...

79 Managing Intranet technology in an organizational context: toward a "stages of growth" model for balancing empowerment and control

Aidan Duane, Pat Finnegan

December 2000 **Proceedings of the twenty first international conference on Information systems**

Full text available: pdf(194.59 KB) Additional Information: full citation, references, citings, index terms

80 Optimal clock period FPGA technology mapping for sequential circuits

Peichen Pan, C. L. Liu

July 1998 **ACM Transactions on Design Automation of Electronic Systems (TODAES)**, Volume 3 Issue 3

Full text available: pdf(252.82 KB) Additional Information: full citation, abstract, references, citings, index terms

We study the technology mapping problem for sequential circuits for look-up table (LUT) based field programmable gate arrays (FPGAs). Existing approaches to the problem simply remove the flip-flops (FFs), then map the remaining combinational logic, and finally put the FFs back. These approaches ignore the sequential nature of a circuit and assume the positions of the FFs are fixed. However, FFs in a sequential circuit can be reposistioned by a functionality-preserving transformation called ...

Keywords: FPGAs, clock period, field-programmable gate arrays, logic replication, look-up tables, retiming, sequential synthesis, technology mapping

Results 61 - 80 of 200 Result page: previous 1 2 3 **4** 5 6 7 8 9 10 next

The ACM Portal is published by the Association for Computing Machinery. Copyright © 2004 ACM, Inc.
Terms of Usage Privacy Policy Code of Ethics Contact Us

Useful downloads: Adobe Acrobat QuickTime Windows Media Player Real Player

PORTAL
US Patent & Trademark Office
Subscribe (Full Service) Register (Limited Service, Free) Login
Search: The ACM Digital Library The Guide
JIT dependency reference
SEARCH
THE ACM DIGITAL LIBRARY
Feedback Report a problem Satisfaction survey

Terms used **JIT dependency reference** Found **13,465** of **134,837**

Sort results by: relevance
Display results: expanded form
Save results to a Binder
Search Tips
Open results in a new window
Try an Advanced Search
Try this search in The ACM Guide

Results 41 - 60 of 200 Result page: previous 1 2 **3** 4 5 6 7 8 9 10 next
Best 200 shown Relevance scale

41 MELDing data flow and object-oriented programming
Gail E. Kaiser, David Garlan
December 1987 **ACM SIGPLAN Notices , Conference proceedings on Object-oriented programming systems, languages and applications**, Volume 22 Issue 12
Full text available: pdf(1.29 MB) Additional Information: full citation, abstract, references, citings, index terms

MELD combines concepts from data flow and object-oriented programming languages in a unique approach to tool reusability. MELD provides three units of abstraction — equations, classes and features — that together allow sufficient options for granularity and encapsulation to support the implementation of reusable tools and the composition of existing tools in parallel (i.e., interleaved) as well as in series.

42 Compilation and run-time systems: DELI: a new run-time control point
Giuseppe Desoli, Nikolay Mateev, Evelyn Duesterwald, Paolo Faraboschi, Joseph A. Fisher
November 2002 **Proceedings of the 35th annual ACM/IEEE international symposium on Microarchitecture**
Full text available: pdf(1.27 MB) Publisher Site Additional Information: full citation, abstract, references, citings, index terms

The Dynamic Execution Layer Interface (DELI) offers the following unique capability: it provides fine-grain control over the execution of programs, by allowing its clients to observe and optionally manipulate every single instruction---at run time---just before it runs. DELI accomplishes this by opening up an interface to the layer between the execution of software and hardware. To avoid the slowdown, DELI caches a private copy of the executed code and always runs out of its own private cache.In ...

43 Session 1C: Interconnect planning: Analysis of substrate thermal gradient effects on optimal buffer insertion
Amir H. Ajami, Kaustav Banerjee, Massoud Pedram
November 2001 **Proceedings of the 2001 IEEE/ACM international conference on Computer-aided design**
Full text available: pdf(132.46 KB) Additional Information: full citation, abstract, references, index terms

This paper studies the effects of the substrate thermal gradients on the buffer insertion techniques. Using a non-uniform temperature-dependent distributed RC interconnect delay model, the buffer insertion problem is analyzed and design guidelines are provided to ensure the near-optimality of the signal performance in the presence of the thermal gradients. In

addition, the effect of temperature-dependent driver resistance on the buffer insertion is studied. Experimental results show that neglect ...

44 Evaluation of the WM architecture

Wm. A. Wulf

April 1992 **ACM SIGARCH Computer Architecture News , Proceedings of the 19th annual international symposium on Computer architecture**, Volume 20 Issue 2

Full text available: pdf(897.06 KB) Additional Information: full citation, abstract, references, citings, index terms

This report describes the results of studies of the WM architecture—its performance, the values of some of its key architectural parameters, the difficulty of compiling for it, and hardware implementation complexity. The studies confirm that, with comparable chip area and without heroic compiler technology, WM is capable of outperforming traditional scalar architectures by factors of 2-9. They also underscore the need to devise higher bandwidth memory systems.

45 Stack inspection: theory and variants

Cédric Fournet, Andrew D. Gordon

January 2002 **ACM SIGPLAN Notices , Proceedings of the 29th ACM SIGPLAN-SIGACT symposium on Principles of programming languages**, Volume 37 Issue 1

Full text available: pdf(318.67 KB) Additional Information: full citation, abstract, references, citings

Stack inspection is a security mechanism implemented in runtimes such as the JVM and the CLR to accommodate components with diverse levels of trust. Although stack inspection enables the fine-grained expression of access control policies, it has rather a complex and subtle semantics. We present a formal semantics and an equational theory to explain how stack inspection affects program behaviour and code optimisations. We discuss the security properties enforced by stack inspection, and also cons ...

46 A controller-based design-for-testability technique for controller-data path circuits

Sujit Dey, Vijay Gangaram, Miodrag Potkonjak

December 1995 **Proceedings of the 1995 IEEE/ACM international conference on Computer-aided design**

Full text available: pdf(149.70 KB) Publisher Site Additional Information: full citation, abstract, references, citings, index terms

This paper investigates the effect of the controller on the testability of sequential circuits composed of controllers and data paths. It is shown that even when both the controller and the data path parts are individually 100\% testable, the composite circuit may not be easily testable by gate-level sequential ATPG. Analysis shows that a primary problem in test pattern generation of combined controller-data path circuits is the correlation of control signals due to implications imposed by the c ...

47 How datapath allocation affects controller delay

Steve C.-Y. Huang, Wayne H. Wolf

May 1984 **Proceedings of the 7th international symposium on High-level synthesis**

Full text available: pdf(559.88 KB) Additional Information: full citation, references, citings

48 Switching activity analysis considering spatiotemporal correlations

Radu Marculescu, Diana Marculescu, Massoud Pedram

November 1994 **Proceedings of the 1994 IEEE/ACM international conference on Computer-aided design**

Additional Information: full citation, abstract, references, citings, index

Full text available: pdf(661.09 KB) terms

This work presents techniques for computing the switching activities of all circuit nodes under pseudorandom or biased input sequences and assuming a zero delay mode of operation. Complex spatiotemporal correlations among the circuit inputs and internal nodes are considered by using a lag-one Markov Chain model. Evaluations of the model and a comparative analysis presented for benchmark circuits demonstrates the accuracy and the practicality of the method. The results presented in this pape ...

49 Provably correct high-level timing analysis without path sensitization

Subhrajit Bhattacharya, Sujit Dey, Franc Brglez

November 1994 **Proceedings of the 1994 IEEE/ACM international conference on Computer-aided design**

Full text available: pdf(820.65 KB) Additional Information: full citation, abstract, references, citings, index terms

This paper addresses the problem of true delay estimation during high level design. The existing delay estimation techniques either estimate the topological delay of the circuit which may be pessimistic, or use gate-level timing analysis for calculating the true delay, which may be prohibitively expensive.We show that the paths in the implementation of a behavioral specification can be partitioned into two sets, SP and UP. While the paths in SP can affect the delay of the circuit ...

50 A parallel, incremental and concurrent GC for servers

Yoav Ossia, Ori Ben-Yitzhak, Irit Goft, Elliot K. Kolodner, Victor Leikehman, Avi Owshanko

May 2002 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2002 Conference on Programming language design and implementation**, Volume 37 Issue 5

Full text available: pdf(231.80 KB) Additional Information: full citation, abstract, references, citings, index terms

Multithreaded applications with multi-gigabyte heaps running on modern servers provide new challenges for garbage collection (GC). The challenges for "server-oriented" GC include: ensuring short pause times on a multi-gigabyte heap, while minimizing throughput penalty, good scaling on multiprocessor hardware, and keeping the number of expensive multi-cycle fence instructions required by weak ordering to a minimum. We designed and implemented a fully parallel, incremental, mostly concurrent colle ...

Keywords: JVM, Java, concurrent garbage collection, garbage collection, incremental garbage collection, weak ordering

51 Interactive constraint-based solid modeling using allowable motion

Mingxian Fa, Terrence Fernando, Peter M. Dew

June 1993 **Proceedings on the second ACM symposium on Solid modeling and applications**

Full text available: pdf(1.01 MB) Additional Information: full citation, references, citings, index terms

Keywords: 3D Interaction Techniques, Constraint-based Solid Modeling, Geometric Reasoning, Virtual Reality

52 Legacy information systems and business process change: a patterns perpective

Ashley D. Lloyd, Dick Dewar, Rob Pooley

December 1999 **Communications of the AIS**

Full text available: pdf(572.94 KB) Additional Information: full citation, references, citings

53 That First Gulp of Java
Brian Christeson, John D. Mitchell
October 1996 **Linux Journal**
Full text available: html(15.64 KB) Additional Information: full citation, abstract, index terms

A relatively new technology, Java has experienced phenomenal growth. Why? Read on

54 Incremental computation via function caching
W. Pugh, T. Teitelbaum
January 1989 **Proceedings of the 16th ACM SIGPLAN-SIGACT symposium on Principles of programming languages**
Full text available: pdf(1.55 MB) Additional Information: full citation, references, citings, index terms

55 FlexiNet—a flexible component oriented middleware system
Richard Hayton, Andrew Herbert, Douglas Donaldson
September 1998 **Proceedings of the 8th ACM SIGOPS European workshop on Support for composing distributed applications**
Full text available: pdf(1.20 MB) Additional Information: full citation, citings, index terms

56 Generating Java trace data
Steven P. Reiss, Manos Renieris
June 2000 **Proceedings of the ACM 2000 conference on Java Grande**
Full text available: pdf(671.64 KB) Additional Information: full citation, references, citings, index terms

57 How java programs interact with virtual machines at the microarchitectural level
Lieven Eeckhout, Andy Georges, Koen De Bosschere
October 2003 **ACM SIGPLAN Notices , Proceedings of the 18th ACM SIGPLAN conference on Object-oriented programing, systems, languages, and applications**, Volume 38 Issue 11
Full text available: pdf(348.88 KB) Additional Information: full citation, abstract, references, index terms

Java workloads are becoming increasingly prominent on various platforms ranging from embedded systems, over general-purpose computers to high-end servers. Understanding the implications of all the aspects involved when running Java workloads, is thus extremely important during the design of a system that will run such workloads. In other words, understanding the interaction between the Java application, its input and the virtual machine it runs on, is key to a succesful design. The goal of this ...

Keywords: Java workloads, performance analysis, statistical data analysis, virtual machine technology, workload characterization

58 A study of locking objects with bimodal fields
Tamiya Onodera, Kiyokuni Kawachiya
October 1999 **ACM SIGPLAN Notices , Proceedings of the 14th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications**, Volume 34 Issue 10
Additional Information:

Full text available: pdf(1.45 MB) full citation, abstract, references, citings, index terms

Object locking can be efficiently implemented by bimodal use of a field reserved in an object. The field is used as a lightweight lock in one mode, while it holds a reference to a heavyweight lock in the other mode. A bimodal locking algorithm recently proposed for Java achieves the highest performance in the absence of contention, and is still fast enough when contention occurs.However, mode transitions inherent in bimodal locking have not yet been fully considered. The algorith ...

59 C and tcc: a language and compiler for dynamic code generation

Massimiliano Poletto, Wilson C. Hsieh, Dawson R. Engler, M. Frans Kaashoek

March 1999 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 21 Issue 2

Full text available: pdf(471.68 KB) Additional Information: full citation, abstract, references, citings, index terms, review

Dynamic code generation allows programmers to use run-time information in order to achieve performance and expressiveness superior to those of static code. The 'C(Tick C) language is a superset of ANSI C that supports efficient and high-level use of dynamic code generation. 'C provides dynamic code generation at the level of C expressions and statements and supports the composition of dynamic code at run time. These features enable programmers to add dynamic code generation ...

Keywords: ANSI C, compilers, dynamic code generation, dynamic code optimization

60 Engineering a customizable intermediate representation

K. Palacz, J. Baker, C. Flack, C. Grothoff, H. Yamauchi, J. Vitek

June 2003 **Proceedings of the 2003 workshop on Interpreters, Virtual Machines and Emulators**

Full text available: pdf(322.87 KB) Additional Information: full citation, abstract, references

The Ovm framework is a set of tools and components for building language runtimes. We present the intermediate representation and software design patterns used throughout the framework. One of the main themes in this work has been to support experimentation with new linguistic constructs and implementation techniques. To this end, framework components were designed to be parametric with respect to the instruction set on which they operate. We argue that our approach eases the task of writing new ...

The ACM Portal is published by the Association for Computing Machinery. Copyright © 2004 ACM, Inc.
Terms of Usage Privacy Policy Code of Ethics Contact Us

Useful downloads: Adobe Acrobat QuickTime Windows Media Player Real Player

Terms used **JIT dependency reference** Found **13,465** of **134,837**

Sort results by: relevance

Display results: expanded form

Save results to a Binder

Search Tips

☐ Open results in a new window

Try an Advanced Search

Try this search in The ACM Guide

Results 21 - 40 of 200 Result page: previous 1 **2** 3 4 5 6 7 8 9 10 next

Best 200 shown Relevance scale

21 Practical extraction techniques for Java

Frank Tip, Peter F. Sweeney, Chris Laffra, Aldo Eisma, David Streeter

November 2002 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 24 Issue 6

properties enforced by stack inspection, and also consi ...

Keywords: Access control, contextual equivalence, equational reasoning, operational semantics, stack inspection

24 HBench:Java: an application-specific benchmarking framework for Java virtual machines

Xiaolan Zhang, Margo Seltzer

June 2000 **Proceedings of the ACM 2000 conference on Java Grande**

Full text available: pdf(775.76 KB) Additional Information: full citation, references, index terms

Keywords: Java performance, benchmarking

25 Using generative design patterns to generate parallel code for a distributed memory environment

Kai Tan, Duane Szafron, Jonathan Schaeffer, John Anvik, Steve MacDonald

June 2003 **ACM SIGPLAN Notices , Proceedings of the ninth ACM SIGPLAN symposium on Principles and practice of parallel programming**, Volume 38 Issue 10

Full text available: pdf(385.41 KB) Additional Information: full citation, abstract, references, index terms

A design pattern is a mechanism for encapsulating the knowledge of experienced designers into a re-usable artifact. Parallel design patterns reflect commonly occurring parallel communication and synchronization structures. Our tools, CO2P3S (Correct Object-Oriented Pattern-based Parallel Programming System) and MetaCO2P3S, use *generative design patterns*. A programmer selects the parallel design patterns that are appropriate for an application, and then adapts the patterns for that specifi ...

Keywords: design patterns, frameworks, parallel programming, programming tools

26 MaJIC: compiling MATLAB for speed and responsiveness

George Almási, David Padua

May 2002 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2002 Conference on Programming language design and implementation**, Volume 37 Issue 5

Full text available: pdf(619.32 KB) Additional Information: full citation, abstract, references, citings, index terms

This paper presents and evaluates techniques to improve the execution performance of MATLAB. Previous efforts concentrated on source to source translation and batch compilation; **MaJIC** provides an interactive frontend that looks like MATLAB and compiles/optimizes code behind the scenes in real time, employing a combination of just-in-time and speculative ahead-of-time compilation. Performance results show that the proper mixture of these two techniques can yield near-zero response time as ...

27 A study of devirtualization techniques for a Java Just-In-Time compiler

Kazuaki Ishizaki, Motohiro Kawahito, Toshiaki Yasue, Hideaki Komatsu, Toshio Nakatani

October 2000 **ACM SIGPLAN Notices , Proceedings of the 15th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications**, Volume 35 Issue 10

Full text available: pdf(225.89 KB) Additional Information: full citation, abstract, references, citings, index terms

Many devirtualization techniques have been proposed to reduce the runtime overhead of

dynamic method calls for various object-oriented languages, however, most of them are less effective or cannot be applied for Java in a straightforward manner. This is partly because Java is a statically-typed language and thus transforming a dynamic call to a static one does not make a tangible performance gain (owing to the low overhead of accessing the method table) unless it is inlined, and partly because t ...

28 As strong as possible mobility (poster session)
Tim Walsh, Paddy Nixon, Simon Dobson
June 2000 **Proceedings of the 22nd international conference on Software engineering**

Full text available: pdf(47.92 KB) Additional Information: full citation, abstract, index terms

An executing thread, in an object oriented programming language, is spawned, directly or indirectly, by a main process. This in turn gets its instructions from a primary class. In Java there is no close coupling of a thread and the objects from which they were created. The use of a container abstraction allows us to group threads and their respective objects into a single structure. A container that holds threads whose variables are all housed within the container is a perfect candidate for ...

29 Decomposable modeling in natural language processing
Rebecca F. Bruce, Janyce M. Wiebe
June 1999 **Computational Linguistics**, Volume 25 Issue 2

Full text available: pdf(921.88 KB) Publisher Site Additional Information: full citation, abstract, references

In this paper, we describe a framework for developing probabilistic classifiers in natural language processing. Our focus is on formulating models that capture the most important interdependencies among features, to avoid overfitting the data while also characterizing the data well. The class of probability models and the associated inference techniques described here were developed in mathematical statistics, and are widely used in artificial intelligence and applied statistics. Our goal is to ...

30 Code optimization II: Code optimization for code compression
Milenko Drinić, Darko Kirovski, Hoi Vo
March 2003 **Proceedings of the international symposium on Code generation and optimization: feedback-directed and runtime optimization**

Full text available: pdf(1.07 MB) Publisher Site Additional Information: full citation, abstract, references

With the emergence of software delivery platforms such as Microsoft's .NET, reduced size of transmitted binaries has become a very important system parameter strongly affecting system performance. In this paper, we present two novel pre-processing steps for code compression that explore program binaries' syntax and semantics to achieve superior compression ratios. The first preprocessing step involves heuristic partitioning of a program binary into streams with high auto-correlation. The second ...

31 Live-structure dataflow analysis for Prolog
Anne Mulkers, William Winsborough, Maurice Bruynooghe
March 1994 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 16 Issue 2

Full text available: pdf(3.59 MB) Additional Information: full citation, abstract, references, citings, index terms, review

For the class of applicative programming languages, efficient methods for reclaiming the memory occupied by released data structures constitute an important aspect of current implementations. The present article addresses the problem of memory reuse for logic programs through program analysis rather than by run-time garbage collection. The aim is

to derive run-time properties that can be used at compile time to specialize the target code for a program according to a given set of queries and ...

Keywords: Prolog, abstract interpretation, compile-time garbage collection, liveness, program analysis

32 Implementing jalapeño in Java

Bowen Alpern, C. R. Attanasio, Anthony Cocchi, Derek Lieber, Stephen Smith, Ton Ngo, John J. Barton, Susan Flynn Hummel, Janice C. Sheperd, Mark Mergen

October 1999 **ACM SIGPLAN Notices , Proceedings of the 14th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications**, Volume 34 Issue 10

Full text available: pdf(1.57 MB) Additional Information: full citation, abstract, references, citings, index terms

Jalapeño is a virtual machine for Java™ servers written in Java.A running Java program involves four layers of functionality: the user code, the virtual-machine, the operating system, and the hardware. By drawing the Java / non-Java boundary below the virtual machine rather than above it, Jalapeño reduces the boundary-crossing overhead and opens up more opportunities for optimization.To get Jalapeño started, a boot image of a ...

33 Parametric polymorphism in Java: an approach to translation based on reflective features

Mirko Viroli, Antonio Natali

October 2000 **ACM SIGPLAN Notices , Proceedings of the 15th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications**, Volume 35 Issue 10

Full text available: pdf(277.16 KB) Additional Information: full citation, abstract, references, citings, index terms

The introduction of parametric polymorphism in Java with translation approaches has been shown to be of considerable interest, allowing the definition of extensions of Java on top of the existing Virtual Machines. Homogeneous translations furthermore, seem to be more useful than heterogeneous, avoiding the continuous increase of library code with redundant information. At this time however, homogeneous approaches aren't as flexible as heterogeneous, with extensions failing to integrate well with ...

34 Quick conservative causality analysis

Ellen M. Sentovich

September 1997 **Proceedings of the 10th international symposium on System synthesis**

Full text available: pdf(965.90 KB) Publisher Site Additional Information: full citation, abstract, references

The causality problem is that of determining if a combinational circuit with cycles has acceptable behavior: that for all inputs the outputs are well defined and stable. While the problem manifests itself at the circuit level, it usually originates at the system level. It may arise when a system is designed as a collection of modules: when composed, cycles are discovered in the ensemble. One must analyze these cycles to correct possible errors or to capture the correct behavior appropriately for ...

Keywords: causality problem, combinational circuit, combinational circuits, conservative algorithm, quick conservative causality analysis, standard logic synthesis techniques

35 Incremental execution of guarded theories

Giuseppe De Giacomo, Hector J. Levesque, Sebastian Sardiña
October 2001 **ACM Transactions on Computational Logic (TOCL)**, Volume 2 Issue 4

Full text available: pdf(245.94 KB) Additional Information: full citation, abstract, references, index terms, review

When it comes to building controllers for robots or agents, high level programming languages like Golog and ConGolog offer a useful compromise between planning-based approaches and low-level robot programming. However, two serious problems typically emerge in practical implementations of these languages: how to evaluate test in a program efficiently enough in an open-world setting, and how to make appropiate nondeterministic choices while avoiding full look ...

Keywords: agent behavior, reasoning about actions, situation calculus

36 Technical correspondence: Assessment of the Java programming language for use in high integrity systems

Jagun Kwon, Andy Wellings, Steve King
April 2003 **ACM SIGPLAN Notices**, Volume 38 Issue 4

Full text available: pdf(319.62 KB) Additional Information: full citation, abstract, references

This paper sets a goal of investigating the use of Java in the development of high integrity systems. Based on previous studies, guidelines, and standards, we develop 23 criteria that are used for the following assessment of Java. A summary of the assessment is provided before we go on to review a few existing subsets of the language.

37 Soot - a Java bytecode optimization framework

Raja Vallée-Rai, Phong Co, Etienne Gagnon, Laurie Hendren, Patrick Lam, Vijay Sundaresan
November 1999 **Proceedings of the 1999 conference of the Centre for Advanced Studies on Collaborative research**

Full text available: pdf(79.70 KB) Additional Information: full citation, abstract, references, citings, index terms

This paper presents Soot, a framework for optimizing Java bytecode. The framework is implemented in Java and supports three intermediate representations for representing Java bytecode: Baf, a streamlined representation of bytecode which is simple to manipulate; Jimple, a typed 3-address intermediate representation suitable for optimization; and Grimp, an aggregated version of Jimple suitable for decompilation. We describe the motivation for each representation, and the salient points in translat ...

38 Confined types

Jan Vitek, Boris Bokowski
October 1999 **ACM SIGPLAN Notices , Proceedings of the 14th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications**, Volume 34 Issue 10

Full text available: pdf(1.71 MB) Additional Information: full citation, abstract, references, citings, index terms

Sharing and transfer of object references is difficult to control in object-oriented languages. Unconstrained sharing poses serious problems for writing secure components in object-oriented languages. In this paper, we present a set of inexpensive syntactic constraints that strengthen encapsulation in object-oriented programs and facilitate the implementation of secure systems. We introduce two mechanisms: confined types to impose static scoping on dynamic object references ...

39 The Performance of Runtime Data Cache Prefetching in a Dynamic Optimization System

Jiwei Lu, Howard Chen, Rao Fu, Wei-Chung Hsu, Bobbie Othmer, Pen-Chung Yew, Dong-Yuan Chen
December 2003 **Proceedings of the 36th Annual IEEE/ACM International Symposium on Microarchitecture**
Full text available: pdf(253.79 KB) Publisher Site Additional Information: full citation, abstract

Traditional software controlled data cache prefetching isoften ineffective due to the lack of runtime cache miss andmiss address information. To overcome this limitation, weimplement runtime data cache prefetching in the dynamicoptimization system ADORE (ADaptive Object code RE-optimization).Its performance has been compared withstatic software prefetching on the SPEC2000 benchmarksuite. Runtime cache prefetching shows better performance.On an Itanium 2 based Linux workstation, it can increasepe ...

40 Technical correspondence: Object serialization analysis and comparison in Java and .NET

Marjan Hericko, Matjaz B. Juric, Ivan Rozman, Simon Beloglavec, Ales Zivkovic
August 2003 **ACM SIGPLAN Notices**, Volume 38 Issue 8
Full text available: pdf(339.96 KB) Additional Information: full citation, abstract, references

This article compares binary and XML object serialization on Java and Microsoft .NET platforms from the performance and size perspective. It uses three different types of objects and different number of objects to make a comparison which reflects real-world circumstances. The article has the following contributions: (1) it compares binary and XML serialization between Java and .NET to compare the efficiency of both platforms; (2) it compares binary and XML serialization within the platforms to c ...

Keywords: .NET, Binary, Java, Serialization, XML

The ACM Portal is published by the Association for Computing Machinery. Copyright © 2004 ACM, Inc.
Terms of Usage Privacy Policy Code of Ethics Contact Us

Useful downloads: Adobe Acrobat QuickTime Windows Media Player Real Player